

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.,)	
)	
Plaintiff,)	
)	
v.)	Civil Action No. 10-03561 WHA
)	
GOOGLE INC.,)	
)	
Defendant.)	

REBUTTAL EXPERT REPORT OF GWYN FIRTH MURRAY

February 8, 2016

TABLE OF CONTENTS

I.	Assignment.....	1
II.	Qualifications.....	1
III.	Summary of Opinions	4
IV.	Open Source Software	5
	A. Open Source Philosophies and Reasons to Use Open Source.....	5
	B. History of FOSS	7
	1) The Free Software Movement	8
	2) The Open Source Initiative (OSI)	11
	C. FOSS vs. “Closed Source” Software	12
V.	FOSS Licenses.....	15
	A. Apache Software License (“ASL”).....	17
	B. The GNU Licenses	17
	1) GNU General Public License (“GPL”).....	17
	2) Lesser GNU General Public License (“LGPL”).....	20
	3) Classpath Exception	22
	C. Compatibility of Licenses.....	22
	D. Enforcement of FOSS Licenses.....	24
VI.	Industry Discussion and Usage of GPL Licenses.....	27
VII.	Counseling Companies on FOSS Compliance	34
VIII.	Background on Java.....	40
IX.	Background on Android	41
X.	Google’s Statements Regarding Potential Inclusion of Copyleft Licenses in Android	43
XI.	Google’s Announcement to Use OpenJDK.....	49
XII.	Anatomy of GPLv2-CE.....	50
XIII.	Google’s Proposed Use of OpenJDK Poses Risks to Android and its OEMs.....	55
	A. Google’s Proposed Use of OpenJDK Code Could Expose Parts of the Android Stack to GPLv2 Obligations.....	57
	B. Google’s Proposed Use of OpenJDK Could Expose Parts of OEMs’ Software to GPLv2 Obligations	60
	C. Hall Fails to Show that Google’s Proposed Combination is Permissible	61
XIV.	Conclusion	62

I. Assignment

1. I have been asked to provide an overview of open source software and open source licensing and to respond to certain opinions of Andrew Hall and Dr. Owen Astrachan concerning open source licenses, including the licensing implications of Google's recent distribution of OpenJDK-based code for Android.
2. In addressing these questions, I draw on my academic and professional background, which includes twenty-eight years of legal practice, the past twenty-six of which have been spent working almost entirely in the high technology industry, and the past seventeen of which have been largely focused on Free and Open Source Software ("FOSS") licensing and compliance.
3. I am being compensated for my work on this case at a rate of \$550 per hour. My compensation is not contingent upon my testimony or upon the result of this proceeding. Appendix A lists the materials I considered in preparing this report.
4. My work is ongoing, and I reserve the right to modify or supplement my conclusions as additional information becomes available to me, or as I perform further analysis.

II. Qualifications

5. I founded the Matau Legal Group in 2002, with a focus on providing the following services:
 - a.) providing general counsel services for small companies that cannot afford, or are not positioned to have, internal counsel;
 - b.) assisting technology clients with drafting and negotiating inbound and outbound licensing and commercial agreements; and
 - c.) providing clients with advice around FOSS licensing and compliance.Since its founding, my law practice has evolved to have a much greater focus on FOSS licensing and compliance: at present, the majority of my clients come to me because of my long history and experience working on FOSS issues, although I often also assist those clients with other issues such as intellectual property licensing and commercial transactions.
6. Examples of the kinds of advice that I provide to my clients around open source matters include:
 - a.) providing training and education on FOSS licensing for in-house legal teams;
 - b.) providing training and education on basics of intellectual property law and FOSS licensing for client engineering and development teams;
 - c.) drafting FOSS compliance policies and processes customized for the particular client, and advising the client on the implementation and maintenance of the same;
 - d.) performing and assisting with third-party

software audits (whether performed manually or via a scan of the client codebase using a third party tool such as Black Duck Software); e.) reviewing the results of such audits, including specifics of which licenses apply to the identified third party software and how the client is using it, and making recommendations for remediation of any noncompliance issues; f.) reviewing proposed FOSS usage, including reviewing software intake questionnaires and meeting with development teams to determine current level of compliance; g.) drafting and reviewing outbound commercial license agreements to ensure that they accurately reflect FOSS license requirements; h.) advising clients on their selection of outbound FOSS license terms, both for clients that are engaged in a dual-licensing model and for those that are launching their own FOSS project or company; and i.) reviewing, revising and negotiating inbound and outbound license and consulting agreements to reflect FOSS usage on behalf of licensors and consultants.

7. I received my Bachelor of Arts from Yale College in 1982, graduating with Distinction in Economics and Magna Cum Laude. I obtained a joint degree from Stanford University in 1987, graduating with a Master of Arts in Latin American Studies and a Juris Doctor degree from Stanford Law School. Following my graduation from Stanford, I worked as an associate attorney for two and one-half years at the Washington, D.C. office of Arnold & Porter, after which I returned to California in 1990 to become an in-house corporate attorney at Apple Computer, Inc. (now “Apple, Inc.”). I left Apple in 1994 to join ALZA Corporation as a corporate attorney, where I remained for one year practicing food and drug law. In 1995, I joined Silicon Graphics, Inc. (later, “SGI”), where I served as Legal Counsel, Americas International and then moved on to become Senior Legal Counsel, Global Services. While I was working in that latter capacity, SGI announced its intention to run its computers on the Linux operating system¹ and I became interested in learning more about Free and Open Source Software.
8. In late 1999, I was offered, and accepted, the position of Director of Legal Affairs at VA Linux Systems, (now “Geeknet”). VA Linux Systems was one of the first companies that built its business model around the use of FOSS. Shortly afterwards, I was promoted to Vice President, Legal Services. During my tenure at VA Linux Systems, where I was exposed to FOSS licenses and the FOSS Community on a daily basis, I was struck by how

¹ <http://www.prnewswire.com/news-releases/sgi-expands-server-operating-system-strategy-company-adds-linux-and-enhances-irix-74548417.html> (last visited Feb. 8, 2016).

few lawyers seemed to know anything about FOSS and its licensing characteristics. I had the idea to create an organization of lawyers practicing in the area of FOSS, that ultimately resulted in my co-founding Open Bar, Inc. Open Bar is a not-for-profit organization founded with the goals of (i) developing clear information about the legal rights and responsibilities of software developers, legal professionals and users of software in the area of FOSS software; and (ii) educating software developers, legal professionals and the general public about the issues, rights and responsibilities associated with the development, use and distribution of FOSS.

9. In early 2001, I left VA Linux Systems to become Vice President – Legal Affairs and General Counsel of Kanisa, Inc. Kanisa was a provider of artificial intelligence customer service solutions and was acquired by Knova in 2007.
10. In 2002, I left Kanisa and founded my current law practice, dba Matau Legal Group.
11. In January 2016, I was invited to join the Open Invention Network, LLC (“OIN”) as a part-time consultant with the title of “Director, OIN Legal Networks.” As per its website at www.openinventionnetwork.com, OIN is a “defensive patent pool and community of patent non-aggression which enables freedom of action in Linux.”² This consultancy position is in addition to my ongoing law practice.
12. I have been admitted to and am a member of the following bar associations: the State Bar of California; District of Columbia Bar (inactive); United States Court of International Trade; District of Columbia Court of Appeals; United States Court of Appeals for the District of Columbia Circuit; and the United States Court of Appeals for the Federal Circuit.
13. I am a member of the following professional organizations: American Bar Association; Santa Clara County Bar Association; Stanford Law School Alumni; Yale Alumni Association; Hunter College High School Alumnae/i Association; The Club Silicon Valley: An Incubator of Women Leaders; Women in Technology International; Ladies Who Lunch Bay Area; FOSS Professionals; SV Forum; Bay Area Open Source Meetup; and the Linux Foundation.
14. I have published various articles on the subject of FOSS, listed in Appendix B.

² <http://www.openinventionnetwork.com/> (last visited Feb. 7, 2016).

15. I have spoken at numerous conferences and presentations relating to FOSS, listed in Appendix C.
16. A copy of my curriculum vitae is attached as Appendix D.

III. Summary of Opinions

17. Free and Open Source Software (“FOSS”) is software whose source code is made “freely” available to be used, copied, modified and distributed. Such software is made available under a variety of licenses, each of which imposes certain requirements and/or conditions on how the software may be used. There are numerous FOSS licenses, including over 75 listed on the Open Source Initiative (“OSI”) website, varying widely in degrees of complexity and formality, and in the permissions governing the use of the licensed code. The interpretation of these conditions and limitations is subject to ongoing debate in the FOSS and legal communities, and FOSS license analysis is often highly context and technology-specific.
18. The GNU General Public License (“GPL”) is a common open source license, and has been released in three separate versions, most recently as version 3 in 2007. All three versions of the GPL are strong “copyleft” licenses, including GPL version 2 (“GPLv2”), which has significant relevance to this case. In general, a “copyleft” license is one that requires that any redistribution of the copyleft code, and any modifications to or works containing the code (or a portion thereof), to be made publicly available and be subject to the requirements of the same copyleft license.
19. The GPLv2 with Classpath Exception (“GPLv2-CE”) is a license that offers a limited “special exception” from the copyleft requirement of GPLv2 for the distribution of *executables*. The Classpath Exception does not provide any exception for distribution of *source code* that would otherwise be required under GPLv2.
20. Although FOSS code is widely used, there remains confusion and disagreement about the meanings and application of various FOSS licenses, including the GPL and GPLv2-CE. Some organizations are wary of using GPL licenses (or any other “copyleft” license) due to the risk that these licenses may limit their ability to maintain their own proprietary code as closed-source, or confidential. This risk is exacerbated by the fact that there is very limited judicial precedent or definitive legal interpretation about what certain provisions of the GPL licenses mean or require.

21. It is my opinion that Google's proposal to include OpenJDK software covered by GPLv2-CE into Android could create a significant risk that additional Android source code, not currently covered by GPLv2-CE, would have to be distributed under that license. In addition, Google's proposal could create significant risk that modifications made by Google's Original Equipment Manufacturers ("OEMs") to Android, which are currently not made public by the OEMs, would also be subject to the GPLv2-CE requirements. As such, I disagree with Mr. Hall (e.g. paragraphs 26-27) that Google's current distribution of OpenJDK-based code in the Android stack would have no impact—from a licensing perspective—on the rest of the stack. And contrary to Mr. Hall's (paragraph 27) and Dr. Astrachan's (paragraph 264) assertions, those same risks would have been present if Google had decided to incorporate OpenJDK-based code in 2007—which of course, they did not do.

IV. Open Source Software

22. The simplest way to define Free and Open Source Software, or "FOSS" is that it is software whose source code is "freely" available to be used, copied, modified and distributed. FOSS software programs may be used for commercial or nonprofit purposes. FOSS software is made available under a variety of licenses that range from "permissive" to "strong copyleft," as discussed in more detail below. FOSS licensing relies on principles of copyright; i.e., in order to release software code as FOSS, the author of that code must hold the requisite copyright and/or license rights that enable it to do so.

A. Open Source Philosophies and Reasons to Use Open Source

23. There are several reasons individuals and companies may choose to release their software under a FOSS license. Within the FOSS community, these reasons may include the beliefs that:

- the public should have the freedom to study, run, copy, modify and redistribute software at will;
- better software is created when more developers have the chance to work on it;
- making software publicly available increases its chance of adoption as a platform;
- credit should be given for developed code (the notion of giving "credit where credit is due"); and
- making software publicly available allows it to be used for altruistic purposes and fosters innovation.

24. Further, many FOSS developers are motivated because they care about and want to enhance their reputations as great programmers – and, perhaps more importantly, because they have a passion for programming and for the particular software. As Chris DiBona has put it:

Much like the rush a runner feels when running a race, a true programmer will feel this same rush after writing a perfect routine or a tight piece of code. . . The point is that many programmers code because it is what they love to do, and in fact it is how they define their intellect.³

25. While the notion of “free” software may seem to conflict with a for-profit business model, FOSS now is firmly entrenched in the for-profit, corporate world, and with good reason. More and more companies are realizing that a route to developing better and more innovative technology, for less money and in less time, is by leveraging FOSS in their product lines and company infrastructures. Many companies are recognizing that FOSS may offer them a way to promote their platforms to facilitate broader adoption of company proprietary technology. Companies are also increasingly realizing that supporting the community of FOSS developers (“the FOSS Community”) can provide significant marketing advantages through increased name recognition and can project the image of the company as a “good corporate citizen.” In other words, engaging with FOSS and the FOSS Community can be a way for commercial entities to open up a broad new range of business and technical opportunities, thereby increasing profits.
26. This said, companies using FOSS in or with their commercial products face some additional challenges not always encountered when using closed-source software. First, given that FOSS code is freely available on the Internet, developers are able to download it at will, and without reviewing the applicable license with their legal counsel. The ability and desire of software engineers to bring software into their companies freely, quickly and without scrutiny raises particular concerns when that software arrives in source code, rather than object code, form. When software comes in as source code, developers’ ability to modify and create derivative works from that software increases. This means that software is easier to commingle with proprietary code – including a company’s “home-grown” code and other third party code. The consequences of such commingling—particularly with respect to FOSS code governed by the “copyleft” license provisions discussed below—can be

³ DiBona, “*Introduction*” in DiBona, Ockman & Stone (editors); Open Sources: Voices from the Open Source Revolution, (O’Reilly 1999) p. 13.

severe. For example, such commingling can trigger requirements to make company and third party source code publicly available upon distribution of the commingled code. Possible consequences include the unintended loss of critical company intellectual property and infringement claims from FOSS or other third party software providers. The risk of such severe consequences often influences companies' decisions whether or not to include FOSS software or make investments related to FOSS software. One of the "worst nightmares" surrounding FOSS is the story of Linksys, which Cisco purchased in 2003 for \$500 million. Three months after the purchase, complaints appeared on FOSS discussion boards claiming that Linksys had violated a key FOSS license by not releasing source code found in its product. Several months and a lot of negotiation later, Cisco publicly released the subject source code and was not able to monetize its investment as it had expected.⁴

B. History of FOSS

27. To understand what FOSS and the FOSS Community are all about, it may be helpful to understand some of their history. The concept of open sharing of information – and indeed of source code – is not new. One commentator has noted that:

Science is ultimately an Open Source enterprise. The scientific method rests on a process of discovery, and a process of justification. For scientific results to be justified, they must be replicable. Replication is not possible unless the source is shared: the hypothesis, the test conditions, and the results. . . Ultimately, the Open Source movement is an extension of the scientific method, because at the heart of the computer industry lies computer science.⁵

28. FOSS started in the early 1980s as a movement among individual computer scientists and software engineers, concentrated at universities or other "think tanks", who proudly called (and call) themselves "hackers." Now, far from being limited to a movement consisting of individual developers, the FOSS movement has grown and includes a wide variety of players, including commercial and non-commercial enterprises, FOSS distributors and service providers and not-for-profit open source projects that foster the development of a particular type of software or FOSS in general.

⁴ See http://www.forbes.com/2003/10/14/cz_dl_1014linksys.html (last visited Feb. 8, 2016); <http://archive.linuxgizmos.com/bruce-perens-deciphering-the-war-on-open-source/> (last visited Feb. 8, 2016); Heather J. Meeker, Open Source and the Legend of Linksys, Jun 28, 2005, Linux Insider, *available at* <http://www.linuxinsider.com/story/43996.html?%20wlc=1300413418>

⁵ DiBona, "Introduction" at 7.

29. Prior to the times when computers and software began to be mass-marketed to the public, most software development occurred in an “open source” environment. Programmers working on large computer systems generally worked together in a collaborative manner on creating code that would run those systems. However, as computer use became more prevalent in the general public, and particularly with the rise of the personal computer and mass-marketing of computer products, the software used to run those computers increasingly was developed as proprietary intellectual property and released in closed-source form. This closed-source code was, and continues to be, licensed under terms that limit its use, copying, modification and distribution – and that usually conditions any or all of these limited rights on the payment of money.

1) The Free Software Movement

30. Richard Stallman, then a researcher at the MIT AI Lab and now well known as the founder of the GNU project⁶ and of its umbrella organization, the Free Software Foundation (“FSF”), believed that the guarding of source code as trade secret property was a threat to innovation. In what he has described as a “stark moral choice,”⁷ Stallman spearheaded the Free Software movement in an effort to make and keep source code available to the general public. Stallman launched the GNU project in 1983,⁸ as a response to his perception that the availability and development of software was being closed off from the public. In his own words, in the early 1980s:

The modern computers of the era, such as the VAX or the 68020, had their own operating systems, but none of them were free software: you had to sign a nondisclosure agreement even to get an executable copy.

This meant that the first step in using a computer was to promise not to help your neighbor. A cooperating community was forbidden. The rule made by the owners of proprietary software was, “If you share with your neighbor, you are a pirate. If you want any changes, beg us to make them.”⁹

31. The GNU project based its operating system design on the Unix design, making it compatible with Unix.¹⁰ GNU is a recursive acronym for “GNU’s not Unix”.¹¹ The

⁶ GNU stands for “GNU’s Not Unix”.

⁷ Stallman, “*The GNU Operating System and the Free Software Movement*”; in DiBona *et al* pp. 53 – 70, at p. 55.

⁸ <https://www.gnu.org/gnu/initial-announcement.html> (last visited Feb. 7, 2016).

⁹ <https://www.gnu.org/gnu/thegnuproject.html> (last visited Feb. 7, 2016).

¹⁰ *Id.*

method that Mr. Stallman chose to ensure that GNU software would remain “free” was to impose license terms governing its use, copying, modification, and distribution. For this reason, he drafted the original GPL.¹² As Mr. Stallman has said:

The goal of GNU was to give users freedom, not just to be popular. So we needed to use distribution terms that would prevent GNU software from being turned into proprietary software. The method we use is called “copyleft” . . . The specific implementation of copyleft that we use for most GNU software is the GNU General Public License, or GNU GPL for short.¹³

32. He believed that “source code is fundamental to the furthering of computer science and freely available source code is truly necessary for innovation to continue.”¹⁴ The Free Software Foundation (the “FSF”) was founded in 1985,¹⁵ with the goals of promoting the freedom to study, create, copy, modify, and distribute computer software. Since then, the FSF has served as the steward of the GNU project, as well as of the GPL licenses. Until 2005, when the Software Freedom Law Center was founded, the FSF was also the primary enforcer of the GPL.

33. As its leading voice, Richard Stallman famously clarified, “the ‘free’ in ‘free software’ refers to freedom, *not price*.” He noted that this “contrasts with the more common proprietary software, which keeps users helpless and divided: the inner workings are secret.”¹⁶

“Free software” means software that respects users’ freedom and community. Roughly, it means that **the users have the freedom to run, copy, distribute, study, change and improve the software**. Thus, “free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free beer”. We sometimes call it “libre software” to show we do not mean it is gratis.¹⁷

¹¹ <http://www.gnu.org/gnu/gnu-history.en.html> (last visited Feb. 7, 2016).

¹² <https://www.gnu.org/gnu/initial-announcement.html> (last visited Feb. 7, 2016).

¹³ <http://www.gnu.org/gnu/thegnuproject.en.html> (last visited Feb. 7, 2016).

¹⁴ <http://www.gnu.org/gnu/thegnuproject.en.html> (last visited Feb. 7, 2016).

¹⁵ <https://www.fsf.org/about/staff-and-board> (last visited Feb. 7, 2016).

¹⁶ <http://www.gnu.org/philosophy/gpl-american-way.en.html> (last visited Feb. 7, 2016) (emphasis in original).

¹⁷ <http://www.gnu.org/philosophy/free-sw.en.html> (last visited Feb. 7, 2016).

34. The FSF defines software as “Free” if the software’s users have the following “four essential freedoms:”¹⁸

- The freedom to run the program as you wish, for any purpose (freedom 0).
- The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

35. As noted above, for software to be considered “Free” or “Open Source” within the FOSS community, its source code must be freely available – and it must be obtainable free of charge. But not all “free” and/or “open source” software meets these requirements. For example, software can be called “free” because it does not cost any money to use it—but source code is not made available, so it is neither “Free” nor “Open Source” software. Or software may be released in source code form at no charge, but with very limited rights on its usage, as may be the case with certain evaluation or beta licenses. Again, such software is neither “Free” nor “Open Source.” And even software source code released into the public domain is not necessarily “Free” software, because it may be incorporated into closed-source software products.¹⁹

¹⁸ *Id.* (emphasis added).

¹⁹ See, for example, comments by Richard Stallman that

If a program is free software when it leaves the hands of its author, this does not necessarily mean it will be free software for everyone who has a copy of it. For example, public domain software (software that has not been copyrighted) is free software; but anyone can make a proprietary modified version of it. Likewise, many free programs are copyrighted but distributed under simple permissive licenses that allow proprietary modified versions.

Stallman, “The GNU Operating System and the Free Software Movement” in DiBona et al, pp. 53-70, at pp. 58-59.

2) The Open Source Initiative (OSI)

36. “Open Source” Software, though similar to “Free Software” for legal purposes, evolved as a result of the 1997 departure of members of the FOSS Community from the Free Software movement who feared that the “viral” characteristics of Free Software would prevent FOSS from being used by commercial enterprises – and, consequently, would limit the growth and adoption of FOSS. They “were concerned that the Free Software Foundation’s anti-business message was keeping the world at large from really appreciating the power of free software.”²⁰ They embarked upon a marketing campaign to promote free software, which led to the development and use of a new term to describe the software they were talking about: “Open Source,” as well as the creation of the Open Source Initiative (the “OSI”) in 1998.²¹
37. “Open Source” Software is similar to “Free Software” in that its licensees are entitled to access, use, copy, modify and distribute OSS source and binary code without making royalty payments to the licensor, and to combine OSS with other software code. And as with “Free Software,” “Open Source” Software means that it is publicly available, but that does not mean that it is always redistributed downstream free of charge. “Open Source” licenses may be “copyleft” licenses, but also may be Permissive licenses that permit the licensee to make his/her modifications to the code private. For this reason, “Open Source” software generally is seen as more commercial-leaning than is “Free Software.”
38. There are various definitions of “Open Source,” but the most referred to is the Open Source Definition (“OSD”) published by the OSI. The OSD clarifies that “[o]pen source doesn’t just mean access to the source code.” Under the OSD, the distribution terms of open-source software must comply with the ten criteria, listed in Appendix E, in order to qualify as “Open Source.”²²
39. The OSI determines whether licenses comply with the OSD. If it determines that they do, they are designated (and marketed) as OSI-approved licenses. The Open Source Initiative currently lists over 75 such “OSI certified” licenses.²³ Though OSI-certification lends

²⁰ DiBona at al., at p. 3.

²¹ <http://opensource.org/history> (last visited Feb. 7, 2016).

²² <https://opensource.org/osd> (last visited Feb. 7, 2016).

²³ See <http://www.opensource.org/licenses/alphabetical> (last visited Feb. 7, 2016).

certain legitimacy to FOSS licenses within the FOSS Community, many other licenses exist that purport to govern the use of free and open source software. In other words, the OSD is not the only definition of what constitutes free and open source software.

C. FOSS vs. “Closed Source” Software

40. Before diving further into particulars of FOSS licensing, it is helpful to consider how FOSS is both similar to, and different from, “closed-source” software. In most respects FOSS and closed-source software licensing are more similar than different. In particular, both types of license depend upon the existence of underlying intellectual property rights—namely, copyright—for their granting and enforcement. There exists confusion in the FOSS community around whether “copyleft” means the absence of intellectual property rights and/or whether “copyleft” is the opposite of copyright. It is not. Copyleft is a play on the word “copyright,” without which there can be no copyleft. Richard Stallman has explained the use of the term as follows:

Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free. The central idea of copyleft is that we give everyone permission to run the program, copy the program, modify the program and distribute modified versions – but not permission to add restrictions of their own. Thus, the crucial freedoms that define “free software” . . . become inalienable rights.²⁴

In other words, “copyleft” licenses require that any code subject to a copyleft license be redistributed only subject to the terms of the same copyleft license. And enforcement of copyleft licenses, as with enforcement of other FOSS and closed-source licenses as well, requires reliance on copyright laws and other intellectual property rights.

41. Open and closed-source software are similar in other ways. For one thing, the development processes for each of these kinds of software consist of common elements. As Paul Vixie has pointed out, the engineering process, whether for closed or open source software, is essentially the same, in that developers need to:
- Identify a requirement, and its requirers;
 - Design a solution that meets the requirement;
 - Modularize the design and plan the implementation²⁵

²⁴ Stallman, “*The GNU Operating System and the Free Software Movement*”; in DiBona *et al* pp. 53-70, at p. 59.

²⁵ Vixie, “Software Engineering”, in DiBona *et al*, pp. 91-100 at p. 99

Mr. Vixie adds an additional requirement: this may or may not happen in the FOSS world, but popular FOSS projects and companies that are writing FOSS often complete his last steps of “Build it; test it; deliver it; support it.”²⁶

42. And, to make money from either kind of software, the same principles of good business strategy apply. Robert Young, a founder of Red Hat Software, has stated:

While making money with free software is a challenge, the challenge is not necessarily greater than with proprietary software. In fact you make money in free software exactly the same way you do it in proprietary software: by building a great product, marketing it with skill and imagination, looking after your customers, and thereby building a brand that stands for quality and customer service.²⁷

43. A major difference between FOSS and “closed source” programs is how the software itself is developed. In the closed-source world, software is developed in secret, by developers working for commercial entities that make money by keeping their software code and other technology confidential, so that they can charge money (royalties) for its use and distribution under commercial license agreements. Accordingly, their developers are bound by employee nondisclosure agreements not to share the software code they create or see with anyone outside the company without their employers’ permission.
44. Given the closed nature of the proprietary development process, while there may be some peer review of software that is developed by and within the company’s own engineering team, there generally is no independent peer review of the software developed within the company. Field testing is carried out through internal testing and some external beta testing, but software generally is not released for commercial use by the intended users until these tests are completed. Support is carried out by internal technical support personnel, or by independent contractors, who are responsible for fixing reported bugs and maintaining the software – all under nondisclosure terms and in exchange for the payment of money.
45. Further, closed-source licenses that permit the use, copying and/or modification of proprietary software typically:

- Make source code unavailable (except under a specific (and likely restrictive) source code license agreement or via a source code escrow arrangement);

²⁶ *Id.*

²⁷ Young; “*Giving It Away: How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry*” in DiBona, et al., pp. 113-125, at p. 114.

- Don't let licensees acquire, install, use, copy or modify software without paying money; and
- Impose downstream restrictions such as prohibitions on sublicensing, copying, redistributing, or modifying the software.

46. Another characteristic of closed-source software licensing is that the applicable license terms usually are written by lawyers and are kept confidential as they are drafted and negotiated. Unless a company's licensee or acquirer requires that it make representations and warranties regarding incoming code, or that the company otherwise deems releasing license terms beneficial for marketing or publicity reasons, the terms on which its software code has been obtained or developed typically are kept secret. Terms may become more public during enforcement actions relating to permitted software use and other license terms but, in most cases, litigation is pursued by individuals or companies through the courts without much public scrutiny. Because license transactions are primarily motivated by revenue, disputes often are resolved through payment of money. And, unless there is a public lawsuit between major players that is resolved via litigation that receives public scrutiny, infringement claims and settlement outcomes are not publicized.

47. In contrast, FOSS programs often are created (and modified) in a collaborative development environment, where individual developers post their contributions of software code as they are created to publicly accessible source code repositories, such as GitHub, Sourceforge, FOSSHub, specific software projects such as Apache or PostgreSQL, or academic sites from which and to which software can be downloaded and uploaded. This software can be accessed and modified by anyone with Internet access, and users who find and fix bugs or make other improvements are able to, encouraged to, and sometimes required to contribute their improvements and modifications back to the software project. This open process of "peer review" throughout the life cycle of FOSS software has led some commentators to the conclusion that "open-source software enjoys the best system-level testing in the industry."²⁸

48. Historically, many FOSS licenses were written via a process similar to that of FOSS development itself. Early FOSS licenses were written primarily by developers themselves, often without the assistance of lawyers, sometimes on their own and sometimes in collaboration with others. Licenses developed in these ways sometimes turn out to be

²⁸ Vixie, "Software Engineering" in DiBona et al, pp. 91-100 at p. 98.

vastly different from the kinds of licenses written by intellectual property and licensing lawyers, including very technical—and sometimes rather quirky—licensing terms.

49. To the extent that FOSS licenses are written by non-lawyer technologists, a number of the early FOSS licenses in particular contain more technical terminology than the software licenses that lawyers may be used to seeing. That said, even the FOSS licenses that have been written either by lawyers or with the involvement of lawyers can be confusing and difficult to interpret.
50. Finally, in the world of FOSS, not only are license terms made public (almost always on the Internet), but members of the FOSS Community take it upon themselves to interpret and enforce those terms. Members of the FOSS Community are committed to the promotion of FOSS, communicate amongst themselves frequently, and see as part of their mission promoting the cause of FOSS and helping enforce license terms. One of the most common introductions to messages or blogs from FOSS Community members about license terms is “IANAL, but. . .” – meaning “I am not a lawyer, but. . .” Long threads on <http://slashdot.org/> and other FOSS message boards are devoted to happenings of interest to the FOSS Community, including notifying others of potential license violations. So, while individual licensors and licensees may raise issues via the courts, many FOSS-related disputes (like the Linksys situation described earlier) have been resolved informally with the “help” or “interference” (depending upon one’s perspective) of active members of the FOSS Community, including the FSF.

V. FOSS Licenses

51. There are three general categories of FOSS licenses: “permissive” or “academic,” “file-based copyleft” or what Mr. Hall calls “weak copyleft,” and those referred to as “copyleft” or “reciprocal.”
52. Before further discussion, it is important to clarify that copyright owners have the full freedom to choose the license or licenses under which they release their code. Copyright holders have exclusive rights to reproduce, make derivative work(s), distribute and license their work, and the rights therefore, to enforce those licenses.
53. Examples of FOSS licenses that are “permissive” or “academic” are the MIT, BSD (an abbreviation for “Berkeley Software Distribution”) and Apache licenses. With the exception of the latest version of the Apache Software License (v 2.0) (the “Apache License”), these are some of the earliest open source licenses, and also are the simplest and

easiest to understand. The essence of the BSD license is found in one line: “*Redistribution and use in source and binary forms, with or without modification, are permitted . . .*”²⁹ In other words, the licensee can essentially do whatever it wants with the licensed code as long as proper license, attribution and warranty disclaimer notices are provided with any distribution. The Apache License, more recently developed, is more complex and is discussed in more detail below.

54. These “permissive” licenses, as applied to the originally licensed code, allow that software code to be used with few restrictions; they permit the licensee to use, copy, create derived works from and distribute the original or derived code under closed-source licenses as well as under FOSS licenses. Accordingly, these licenses are used frequently by commercial enterprises, often with the result that the code is “forked” and derivative works are lost to the FOSS community. Given their relative lack of restrictions on downstream use and distribution, however, many permissive licenses also are compatible with other FOSS licenses and are widely used by the FOSS Community.
55. A second category of FOSS licenses are referred to as “file-based copyleft,” which Mr. Hall refers to as “weak copyleft” licenses. I do not agree with Mr. Hall’s characterization of these licenses as “weak copyleft,” and I prefer to use the term “file-based copyleft” as a more accurate description. These licenses require that derivative works of the licensed work, if distributed, be distributed under the terms of the same license – in this regard, these are strong copyleft licenses. However, these licenses often also explicitly except out additions to the licensed work that are not modifications of the originally-licensed code itself from the copyleft provisions of the license.
56. Some of the better known of these licenses include the Mozilla Public License, and the Eclipse Public License (“EPL”) (which is a modified version of the Common Public License). In the case of the EPL, for example, a “Contribution” (which would be subject to copyleft requirements under the license) does “not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.”³⁰ The EPL permits relicensing of EPL-licensed code in executable form under the terms of a different

²⁹ See <http://www.opensource.org/licenses/bsd-license.php> (last visited Feb. 7, 2016).

³⁰ EPL Section 1(b) (www.eclipse.org/legal/epl-v10.html).

license, provided that the licensor makes the originally EPL-licensed code available in source code form.

57. Strong copyleft licenses include the GNU family of licenses, including the various versions of the GPL, which require that any GPL-licensed code and derivative works thereof be distributed – and redistributed – under the same GPL license terms. I discuss these licenses in more detail below.

A. Apache Software License (“ASL”)

58. As noted above, the Apache License is a permissive license, in that it does not require that either the originally-licensed code or modifications thereto be distributed under the same license. The Apache License does, however, impose obligations on licensees with respect to maintenance of the copyright, patent, trademark, and attribution notices in redistributed code, and also requires that notice be given if modifications are made to the code. In addition, the Apache License includes explicit patent license and patent defense terms that other permissive licenses such as the BSD and MIT licenses do not. The Apache Software Foundation (“ASF”) adopted the current version of the Apache License, version 2.0, in January 2004. The ASF’s stated goals in revising the license included making the license easier for non-ASF projects to use, improving compatibility with GPL-based software,³¹ allowing the license to be included by reference instead of listed in every file, clarifying the license terms related to the submission of contributions and requiring a patent license on contributions that necessarily infringe a contributor’s own patents.³²

B. The GNU Licenses

1) GNU General Public License (“GPL”)

59. The GNU General Public License, or GPL, is one of the most widely used FOSS licenses³³ and was originally drafted by Richard Stallman. There are three versions of the GPL: a.) version 1.0, which is no longer in use; b.) version 2.0, which remains widely used and has particular significance for this case; and c.) version 3.0, which was released in 2007. All three GPL versions are strong copyleft licenses, in that they require that any distribution of

³¹ However, as noted below, the FSF has stated that certain Apache licenses are not compatible with some of the GPL licenses.

³² <http://www.apache.org/licenses/> (last visited Feb. 7, 2016).

³³ <https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses> (last visited Feb. 8, 2016).

the licensed code—and any derivative works thereof, or “works based on the Program”—be made under the terms of the same license. (A slight exception to this last point is that sometimes the GPL is applied to covered software with the notation “GPL v.2 or later” or “GPL v.3 or later version,” thus permitting the license to migrate to a later version.)

60. Section 2 of GPLv2 provides, in relevant part, that:

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work . . . provided that you also . . . cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

61. Whether the GPL’s “work based on the Program” equates to what is a “derivative work” under U.S. copyright law, and whether the GPL’s terminology expands upon copyright law’s notion of a “derivative work,” is the subject of much discussion in the FOSS legal community.

62. 17 U.S.C. Section 101 defines a “derivative work” as follows:

“A ‘derivative work’ is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a ‘derivative work.’”³⁴

63. Among the questions under ongoing discussion in the FOSS legal community is the extent to which linking a GPL’d Program to an independent module creates a derivative work under copyright law, or a “work based on the Program” under the GPL, and whether these should be the same, or must be separate, lines of inquiry. On the question of whether linking to GPL’d code creates a “work based on the Program,” the FSF takes the position that it does:

[Q.] You have a GPL’ed program that I’d like to link with my code to build a proprietary program. Does the fact that I link with your program mean I have to GPL my program?

³⁴ “Collective work” on the other hand is a work, such as a periodical issue, anthology, or encyclopedia, in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole.

[A.] Yes.³⁵

64. However, not all agree with the FSF's position. Linus Torvalds, the creator of the original Linux code, and who is still active in its maintenance and improvement, has stated:

The GPL requires that works "derived from" a work licensed under the GPL also be licensed under the GPL. Unfortunately what counts as a derived work can be a bit vague. As soon as you try to draw the line at derived works, the problem immediately becomes one of where do you draw the line?

We ended up deciding (or maybe I ended up decreeing) that system calls would not be considered to be linking against the kernel. That is, any program running on top of Linux would not be considered covered by the GPL. This decision was made very early on and I even added a special read-me file (see Appendix B) to make sure everyone knew about it. Because of this commercial vendors can write programs for Linux without having to worry about the GPL.³⁶

65. The Exhibit B to which Mr. Torvalds refers above is a copy of the preamble to the GPL that Mr. Torvalds added to the Linux kernel code, and reads:

Note! This copyright does **not** cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does **not** fall under the heading of "derived work." Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.³⁷

66. Thus, the author and copyright holder of the GPL licenses, the FSF, has offered a very different interpretation of what constitutes a derivative work than that of the original author and copyright holder of Linux itself. This difference points out that there exists ongoing confusion and disagreement about the weight that should be given to the intent of the licensor when interpreting GPL licenses, what constitutes a derivative work or "work based on the Program" under these licenses, among other issues.

³⁵ <http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.en.html#TOCLinkingWithGPL> (last visited Feb. 8, 2016).

³⁶ Torvalds, Linus, "The Linux Edge", in DiBona pp. 108-109.

³⁷ *Id.* at 263. Note, that this notice also appears in the Linux kernel code itself. <https://www.kernel.org/pub/linux/kernel/COPYING> (last visited Feb. 8, 2016).

2) Lesser GNU General Public License (“LGPL”)

67. The LGPL was first published in 1991, and given the version number 2 so as to be on par with GPLv2. In its first version, it was called the “Library General Public License.” The LGPL was revised and re-published as version 2.1 in 1999, when it was renamed the “GNU Lesser General Public License” because the FSF considered that it did “[l]ess to protect the user’s freedom than the ordinary General Public License.”³⁸ Version 3 of the LGPL was published at the same time as GPLv3, in 2007. Version 3 of the LGPL is structured differently: Rather than being a separate license in itself, it was published as an amendment to GPLv3 rather than as a standalone document.
68. The LGPL was created to permit the linking of LGPL libraries to some “non-free” programs, as the FSF explains in the Preamble to version 2.1 of the license:

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

69. As with the GPL, there is significant controversy in the FOSS legal community about the interpretation of LGPL provisions. Discussions around LGPL focus on the extent to which the LGPL actually differs from the GPL in its copyleft implications, as well as to what its terms “work that uses the Library” vs. “work based on the Library” encompass. For example, one commentator has said that:

. . . a careful comparison of the text of the GPL and LGPL licenses. . . reveals that, if the process of adding or deleting library functions creates a *derivative work* of the Library, then the LGPL functions identically to the GPL.³⁹

And goes on to state:

³⁸ LGPLv2.1, Preamble (*see* <http://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html>).

³⁹ Rosen, Larry, Open Source Licensing: Software Freedom and Intellectual Property Law (2005), pp. 124. (emphasis in original)

The LGPL, therefore, is an anomaly – a hybrid license intended to address a complex issue about program linking and derivative works. It doesn't solve that problem but merely directs us back to the main event, the GPL license itself.⁴⁰

70. Section 5 of the LGPLv2 provides, in relevant part:

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library.” Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.⁴¹

71. While the LGPL does permit linking of a “work that uses the Library” to an LGPL'd library to create an executable, it places additional requirements on the distribution of that resulting executable that are often looked over. For example, Section 6 of the license provides:

[Y]ou may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

72. These terms limiting the prohibition of reverse engineering are not consistent with those of most closed-source licenses. As Heather Meeker points out, “[this language] is not consistent with the terms of most end user licenses, which restrict reverse engineering entirely, so care must be taken when integrating LGPL libraries with proprietary programs.”⁴²

73. Further, like the GPLv2, the LGPLv2.1 requires that any modifications to the original LGPL code be made available in source code form upon distribution of that code or modifications to that code.

⁴⁰ *Id.* at p. 124-125.

⁴¹ Emphasis added.

⁴² Meeker at 40.

74. . And, like the GPLv2, the LGPLv2.1 requires that any modifications to the LGPL code be made available in source code form upon distribution of that code or those modifications.

3) Classpath Exception

75. GPLv2 with Classpath Exception (the “GPLv2-CE”) is, as its name suggests, the GPLv2 with a special exception that applies in limited circumstances. The Classpath Exception states that:

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

This language reaffirms the provisions of the GPL itself.

76. The Classpath Exception then goes on to describe a limited exception to the above statement:⁴³

As a special exception, the copyright holders of this library give you permission to link this library with **independent modules to produce an executable**, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. **An independent module is a module which is not derived from or based on this library.** If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

77. While both the LGPLv2.1 and GPLv2-CE provide certain exceptions to the full copyleft implications of the GPLv2 license, each of these licenses maintain the strong copyleft provisions of the GPL when it comes to the distribution of modifications or “works based” on the original code. For this reason, I disagree with Mr. Hall’s characterization of these two licenses as “weak copyleft” licenses.

C. Compatibility of Licenses

78. Organizations may—and often do—seek to combine software elements released under different licenses into a single offering or relicense them on different terms. However, the decision to combine different licenses and the way in which the combination is implemented must be thought through carefully in order to avoid unintended

⁴³ <http://www.gnu.org/software/classpath/license.html> (last visited Feb. 7, 2016) (emphasis added).

consequences. The notion of “license compatibility” is not unique to FOSS licensing. For example, if a closed-source software vendor licenses its software to another company with a restriction on relicensing the software, the licensee company is not permitted to grant a sublicense to that software to a third party. Similarly, if a company wishes to include GPL’d or other copyleft-licensed software in its software product, and then relicense that software under a “closed source” license, it is not permitted to do so. In each such scenario, the inbound and outbound licenses are “incompatible.” Among FOSS licenses, there are similar issues of incompatibility among certain licenses.

79. For example, the ASF’s website⁴⁴ lists licenses that “may not be included within Apache products.” These include both “GNU LGPL 2, 2.1, 3” and “Special exceptions to the GNU GPL (e.g. GNU Classpath). The explanation for each of these is given below:

GNU LGPL

The LGPL is ineligible primarily due to the restrictions it places on larger works, violating the third license criterion. Therefore, LGPL-licensed works must not be included in Apache products.

Special exceptions to the GNU

Some copyright holders have licensed their works under the GPL with special exceptions. Although these exceptions may appear to be addressing the restrictions disallowed by the ASF’s first and second license criteria, the exceptions may only apply to software not “derived from or based on” the covered work. This references terms defined in the GPL that include works that “use” or “contain” the work.

80. However, the ASF acknowledges that there is still uncertainty with respect to certain licenses. On a separate page,⁴⁵ the compatibility of Apache License 2.0 with GPLv2 is specifically addressed. The passage indicates that while Apache tried to make its license compatible with GPL, the Free Software Foundation believes the licenses are not compatible:

Despite our best efforts, the FSF has never considered the Apache License to be compatible with GPL version 2, citing the patent termination and indemnification provisions as restrictions not present in the older GPL license. The Apache Software Foundation believes that you should always try to obey the constraints expressed by the copyright holder when redistributing their work.

⁴⁴ <http://www.apache.org/legal/resolved.html> (last visited Feb. 7, 2016).

⁴⁵ <http://www.apache.org/licenses/GPL-compatibility.html> (last visited Feb. 7, 2016).

81. As such, on the GNU website, Apache License 2.0 is listed as *incompatible* with GPL version 2:

Please note that this license is not compatible with GPL version 2, because it has some requirements that are not in that GPL version. These include certain patent termination and indemnification provisions. The patent termination provision is a good thing, which is why we recommend the Apache 2.0 license for substantial programs over other lax permissive licenses.⁴⁶

D. Enforcement of FOSS Licenses

82. From the launch of the FSF until 2007, FOSS license compliance in the U.S. was undertaken mainly through informal channels – whether via an online discussion group in an Internet forum, or by the FSF. The FSF’s General Counsel, Eben Moglen, has explained why informal threats of bad publicity were effective in promoting compliance with the GPL licenses:

. . .no one wanted to be seen as the villain who stole free software, and no one wanted to be the customer, business partner, or even employee of such a bad actor. Faced with a choice between compliance without publicity or a campaign of bad publicity and a litigation battle they could not win, violators chose not to play it the hard way.⁴⁷

83. The Cisco/Linksys situation mentioned earlier initially came to light in the course of discussions within the developer community. In an article discussing FOSS license enforcement,⁴⁸ Heather Meeker discussed how after Cisco’s acquisition of Linksys, in June 2003, “complaints appeared on discussion boards such as LKML⁴⁹ and Slashdot claiming that Linksys was violating the GPL by not providing source code for certain code used in its WRT54G wireless access point.” Thereafter, the FSF sought to enforce the terms of the

⁴⁶ <http://www.gnu.org/licenses/license-list.en.html#apache2> (last visited Feb. 8, 2016).

⁴⁷ Moglen, “Enforcing the GNU GPL”; (10 September 2001), located at <http://www.gnu.org/philosophy/enforcing-gpl.html> (last visited Feb. 1, 2016).

⁴⁸ Heather J. Meeker, *Open Source and the Legend of Linksys*, Jun 28, 2005, Linux Insider, available at <http://www.linuxinsider.com/story/43996.html?%20wlc=1300413418>.

⁴⁹ <https://lkml.org/lkml/2003/6/7/164> (last visited Feb. 8, 2016).

GPL through direct discussions with Cisco which resulted in the release of the Linksys source code at issue three to four months after the first demand from FSF.⁵⁰

84. Similarly, in Europe, the majority of GPL enforcement actions have been informal, initiated by groups like gpl-violations.org. The founder of gpl-violations.org, Harald Welte, also filed a few formal court actions to enforce the provisions of the GPL as they applied to code that he personally had written and in which he held the copyrights.⁵¹ These court actions resulted in, *inter alia*, court decisions confirming the enforceability of the GPL but that provided little other discussion of the GPL itself. However, the results of the informal inquiries of GPL violations of Mr. Welte and gpl-violations.org have not been widely publicized.⁵²

85. Formal actions in the United States to enforce the GPL started in 2007, after the founding of the Software Freedom Law Center (SFLC) was launched in 2005 to promote enforcement of FOSS licenses and to provide pro bono legal services for open source software projects.⁵³ In addition to its license enforcement activities, the SFLC provides FOSS license development efforts, trademark counseling, public education, consulting and training, patent defense and license defense services to its clients.⁵⁴

86. The SFLC filed its first lawsuit in 2007 on behalf of the copyright holders of BusyBox software, Mr. Andersen and Rob Landley, against Monsoon Multimedia, alleging its violation of GPLv2, under which license BusyBox had been released.⁵⁵ The SFLC followed

⁵⁰ http://www.forbes.com/2003/10/14/cz_dl_1014linksys.html (last visited Feb. 7, 2016); <http://archive.linuxgizmos.com/bruce-perens-deciphering-the-war-on-open-source/> (last visited Feb. 7, 2016).

⁵¹ Jennifer Buchanan O'Neill and Christopher J. Gaspar, What Can Decisions by European Courts Teach Us About the Future of Open-Source Litigation, 38 AIPLA Q.J. 437 (2010); Kevin J. O'Brien, In open source, an unexpected trap License rules snare even big tech firms, December 10, 2005, <https://www.highbeam.com/doc/1P1-116284909.html>

⁵² <http://www.linuxinsider.com/story/43996.html?%20wlc=1300413418> (last visited Feb. 8, 2016).

⁵³ <http://www.groklaw.net/articlebasic.php?story=20050201060017590> (last visited Feb. 7, 2016).

⁵⁴ <https://www.softwarefreedom.org/services/> (last visited Feb. 7, 2016).

⁵⁵ Steven Vaughn-Nichols, *First Lawsuit Filed Over Alleged GPL Violation*, ExtremeTech.com (2007).

this lawsuit with several other suits on behalf of the BusyBox developers alleging similar violations of the GPL,⁵⁶ including those against Xterasys Corporation, High-Gain Antennas and Verizon Communications in 2007, and Bell Microproducts, Inc., Super Micro Computer, Inc. in 2008.⁵⁷

87. By the end of 2009, the SFLC, on behalf of the Software Freedom Conservancy⁵⁸ had filed suit against fourteen large corporations, including Samsung, Best Buy, Westinghouse and Robert Bosch LLC, alleging illegal copying and distributing of the Busybox code.⁵⁹ Most of these lawsuits also resulted in settlements⁶⁰ whereby the BusyBox plaintiffs agreed to dismiss the lawsuits and to reinstate the defendants' rights to distribute BusyBox under the GPL. In return, defendants agreed to meet certain conditions which included, by way of example, (i) appointing an open source compliance officer within its organization to monitor and ensure GPL compliance, (ii) publishing the source code for the version of BusyBox it previously distributed on its website, and (iii) undertaking efforts to notify third

⁵⁶ Charles Babcock, *Extreme Networks Latest Target of GPL Enforcement*, July 22, 2008, <http://www.informationweek.com/software/operating-systems/extreme-networks-latest-target-of-gpl-enforcement/d/d-id/1070196?>

⁵⁷ Beth Z. Shaw, *Recent Lawsuits Reflect Open Source Software Users' Copyright Compliance Obligations*, Mondaq, May 7, 2010.

⁵⁸ <https://sfconservancy.org/news/2006/apr/03/launch/>; <https://sfconservancy.org/about/> (Non-profit organization launched in 2006 with the help of SFLC to provide FOSS developers with free services and enforce GPL compliance.)

⁵⁹ Beth Z. Shaw, *Recent Lawsuits Reflect Open Source Software Users' Copyright Compliance Obligations*, Mondaq, May 7, 2010; <http://www.zdnet.com/article/best-buy-samsung-westinghouse-11-more-named-in-gpl-lawsuit/>

⁶⁰ <https://www.softwarefreedom.org/news/2010/jun/07/motion-against-westinghouse-digital-electronics-gp/> (As of this date, four defendants (Samsung, Comtrend, Dobbs-Stanford, and GCI Technologies) have settled with the plaintiffs.); <http://brownrudnick.com/blog/emerging-technologies/license-revoked-applying-section-4-of-the-gpl-and-the-lessons-of-best-buy-to-googles-android/>; <http://www.groklaw.net/pdf3/SFCvBestBuy-164.pdf>

As of the filing of this motion, Plaintiffs have reached settlement with nine of the fourteen defendants and received a default judgment from the court with respect to one other defendant. Thus, only four defendants remain. Settlement discussions with two of those defendants have been very productive and plaintiffs believe settlement with them is imminent. Plaintiffs have hoped to reach an amicable settlement with the remaining two defendants, Best Buy and Phoebe Micro, however, settlement negotiations with those two defendants have proven futile.

parties to which they had distributed Busybox of their rights to the software under the GPL.⁶¹

88. Despite the fact that there now have been multiple formal enforcement actions filed in U.S. courts in recent years, almost all of these actions have been settled out of court -- with only limited information about the settlement terms being made available to the public. And, aside from the few court rulings in Europe that essentially determined that the GPL was an enforceable license, without providing significant guidance as to its scope or interpretation, there remains little judicial precedent in the US or elsewhere that assists lawyers in the FOSS community with interpreting the terms of the GPL and other licenses.

VI. Industry Discussion and Usage of GPL Licenses

89. While GPL is a commonly used open source software license, its scope and meaning are rarely the subject of litigation. In fact, up until now, I am not aware of any litigation resolving the scope and interpretation of either the GPL licenses themselves, or of the Classpath Exception.

90. To the contrary, there is ample evidence that confusion and uncertainty about the meaning and impact of the GPL's copyleft terms is widespread in the industry. As an example, Michael Morgan, in his academic article examining the "viral aspects" of the GPL, he states that historically "the GPL has been a *difficult document to understand*."⁶²

91. As a further example, Bill Burke, who was Chief Architect at JBoss, before the company was acquired by Red Hat,⁶³ wrote a blog in 2010 about the different GPL licenses and argued that "the distinctions between GPL, LGPL, and ASL [Apache Software License] are pretty much meaningless [sic] to most consumers of OSS."⁶⁴

⁶¹ <http://www.softwarefreedom.org/news/2007/oct/30/busybox-monsoon-settlement/>;
<http://www.softwarefreedom.org/news/2008/mar/06/busybox-hga/>;
http://www.liquisearch.com/busybox/gpl_lawsuits;
<http://www.softwarefreedom.org/news/2008/jul/23/busybox-supermicro/>

⁶² Michael F. Morgan, The Cathedral and the Bizarre: An Examination of the "Viral Aspects" of the GPL, 27 J. MARSHALL J. COMPUTER & INFO. L. 349, at 351 (2010) (emphasis added).

⁶³ <http://bill.burkecentral.com/about-me/> (last visited Feb. 6, 2016).

⁶⁴ <http://bill.burkecentral.com/2010/05/19/apache-damaging-to-open-source/> (last visited Feb. 4, 2016).

92. Thereafter, Jilles Van Gurp, Principal Engineer of Nokia, disagreed with the blog post and explained:

I work for a big company (Nokia) that is an active contributor in many OSS communities. **Dealing with the viral nature of the GPL is as far as I know a royal pain in the ass for any big company.** Like it or not, big companies have a huge investment in legacy code. Nokia actually went through the effort of open sourcing a lot of their code recently (i.e. Symbian), of which we are rightly proud. This was a major undertaking and a big investment. **Since this is an industry with many players and vested interests, Nokia chose a business friendly license: the Eclipse Public License.** The EPL is very similar in nature to the Apache License. Both Apache and Eclipse are well respected and widely used OSS licenses. Both are business friendly in the sense that they are non viral and low risk.^[65] **Our legal people require lots of checks and balances before we use GPLed/IGPLed code because the potential for accidental infringement is huge and mostly not considered worth the trouble. Checks for EPL & Apache licensed code are a lot more relaxed.**

Both the Eclipse and Apache licenses are widely used in some of the largest OSS communities in the Java world, e.g. Spring, Eclipse, Android, Apache Harmony, Tomcat, Geronimo, etc. All these communities have strong industry participation with regular contributions from major players in the industry. If I look at JBoss I see a community where most of the committers are Red Hat employees and a decline in popularity that has been going on for several years now. **The type of companies that could reasonably be expected to be interested in committing on Jboss are instead (very) active in the apache and eclipse community. I'm pretty sure license and IPR considerations are a big part of that story.**

Licensing is a complex issue and I agree it is best to be pragmatic about it. But from community point of view it matters. If you want to build a community around your software, Apache and Eclipse style licensing is going to make it a lot easier to get big corporations on board. If you are not interested in community building, the license doesn't matter that much either. If I look at how big corporations have actually been using GPL3 I cannot but include it is being used because of its virality. E.g. Sun has wielded dual licensing, GPL 2 & 3 and the **so-called classpath exception to ensure most depending companies come to them for a commercial license.** There's very little of that going on in the Apache and Eclipse communities."⁶⁶

⁶⁵ Note that I do not agree with Mr. Burke's characterization of the Eclipse Public License as "non viral." As discussed earlier, I consider that license a strong copyleft license, but one that contains a clear exception from its copyleft requirements with respect to "do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program."

⁶⁶ *Id.* (emphasis added).

93. This discussion is just one example of many that have taken place in the FOSS community highlighting the debate and confusion about the meaning of the various GPL licenses. Discussions on the Stack Overflow website (a popular website for developers) demonstrate that confusion abounds with respect to the GPLv2-CE itself:

Oracle seems to license all their Java-related open source code under the GPL with a Classpath Exception.^[67] From what I understand, this seems to allow to combine [sic] these libraries with your own code into products that do not have to be covered by the GPL. How does this work? What are examples of how I can and cannot use these classes? Why was this new license used as opposed to the LGPL, which seems to allow for pretty much the same things, but is better established and understood? What are the differences to the LGPL?

The various responses to this post illustrated a general lack of consensus on the subject:

“I found this on wikipedia: en.wikipedia.org/wiki/... which clears things up a little, but I still don’t understand why someone would use this over LGPL.”

“That Wikipedia pages does not mention things like ‘user must be able to swap out library with his own version’ which are provisions of the LGPL. So maybe GPL+Exception does not require that?”

“Reading on, that does seem to be the case: ‘LGPL formulates more requirements to the linking exception: you must allow modification of the portions of the Library you use and reverse engineering (of your program and the library) for debugging such modifications. In this case, what am I not allowed to do under this license?’”

At one point, a respondent suggests: “Shouldn’t you be talking to a lawyer about this?”⁶⁸

94. Even the authors of the GPL licenses have acknowledged that there is a lack of understanding of how to comply with the requirements of the GPL. The Software Freedom Law Center (SFLC, which represented and advised the Free Software Foundation throughout the process of drafting of GNU General Public License v3)⁶⁹ has acknowledged

⁶⁷ This, by the way, is not entirely correct. In particular, OpenJDK Java Virtual Machine is subject to the GPL without Classpath Exception. <http://openjdk.java.net/faq/> (last visited Feb. 8).

⁶⁸ <http://programmers.stackexchange.com/questions/119436/what-does-gpl-with-classpath-exception-mean-in-practice> (last visited Feb. 4, 2016).

⁶⁹ <http://www.softwarefreedom.org/news/2008/jan/19/gplv3-support/> (last visited Jan. 29, 2016).

that this is due to a basic lack of understanding of how to comply. In an article in 2008 Bradley M. Kuhn et. al. discuss this point:⁷⁰

However, in our experience with GPL enforcement, few redistributors' compliance challenges relate directly to the copyleft provisions; this is doubly true for most embedders. Instead, the distributions of GPL'd systems that we encounter typically consist of a full operating system including components under the GPL (e.g., Linux, BusyBox) and components under the LGPL (e.g., the GNU C Library). Sometimes, these programs have been patched or slightly improved by direct modification of their sources, resulting unequivocally in a derivative work. Alongside these programs, companies often distribute fully independent, proprietary programs, developed from scratch, which are designed to run on the FOSS operating system but do not combine with, link to, modify, or otherwise derive from the GPL'd components. In the latter case, where the work is unquestionably a separate work of creative expression, no derivative work has been created. The tiny minority of situations which lie outside these two categories, and thus involve close questions about derivative works, require a highly fact-dependent analysis and cannot be addressed in a general-purpose document. Most companies accused of violations, however, lack a basic understanding of how to comply even in the straightforward scenario.⁷¹

95. Heather Meeker has discussed companies' increased concern about the possibility of their own noncompliance with inbound FOSS licenses. She describes how, over time,

[The focus of risk analysis in open source has categorically moved from infringement risk to compliance risk. In other words, most companies today are not so concerned that the open source code they get from the community is infringing so much as the risk that their own particular use of open source code does not comply with the requirements of the open source license applied to the code.⁷²

96. Given the uncertainty about the scope and interpretation of the various GPL licenses, some industry players have moved towards more permissive licensing models. For example, "Alfresco, SpringSource, Signavio and Camunda have launched an open source project under the permissive Apache 2.0 license, spawned mainly by prospective Alfresco OEM

⁷⁰ <https://www.softwarefreedom.org/resources/2008/compliance-guide.html> (last visited Jan. 29, 2016).

⁷¹ *Id.*

⁷² Heather Meeker, "Open Source and the Secrets of Commando Due Diligence," 44 Tex. J. of Bus. L. 561, 562.

partners' weariness of LGPL software."⁷³ It seems that Alfresco moved from LGPL to a more permissive license to address the concerns of larger software companies who are concerned about any license with "the letter G in it." Alfresco CTO John Newton stated:

Activiti emerged from our desire to have an Apache-licensed BPM engine. Although we were quite happy with the (RedHat JBoss) jBPM engine, it's [sic] LGPL license was preventing us from OEMing Alfresco to larger software companies that were concerned about any open source license with the letter G in it. It's irrelevant that they shouldn't be concerned about it . . .⁷⁴

97. A survey was conducted by OpenLogic (a company that provides solutions for Open Source Software) in 2007 to determine the frequency with which companies were using various open source licenses. The survey found that of the top 20 open-source projects being run by its customers, 75% were licensed under the Apache Software License, whereas only 20% were licensed under "GPL or LGPL." The study found that larger projects (by number of customers) were less likely to use "GPL or LGPL" than smaller projects.⁷⁵

98. OpenLogic released another study in 2011 showing similar results with the Apache License being the most commonly chosen license among enterprises. Kim Weins, senior vice president of marketing at OpenLogic commented on this trend:

This research reveals a license divide between open source developers and enterprises. Hands down, developers prefer using the GPL license for their code, but **enterprises tend to avoid the GPL in favor of the Apache license or other liberal licenses.** When you examine the data from enterprises using the OpenLogic Deep Discovery scanning tool, you are primarily capturing the use of open source by companies distributing software or electronics products containing software. These companies often try to avoid using open source licenses due to concerns about copyleft requirements and the impact on their own intellectual property.

The take-away point is that simply counting the number of open source projects under a particular license does not tell the whole picture. These statistics indicate that GPL advocates have more work to do to convince enterprises to embrace the GPL. At the same time, they show that open

⁷³ <http://www.infoworld.com/article/2626845/technology-business/activiti-bpm-project-questions-value-of-lgpl-gnu-license.html> (last visited Jan. 28, 2016).

⁷⁴ *Id.*

⁷⁵ <http://www.openlogic.com/resources/enterprise-blog/archive/busting-the-myth-of-gpl-dominance-apache-rules> (September 27, 2007) (last visited Jan. 28, 2016).

source developers choosing more liberal licenses will lower the barriers to enterprise adoption.⁷⁶

99. Confirming these findings, a study by Stephen O’Grady, co-founder of Red Monk, a major research group, observed a clear move away from GPL licenses towards more permissive licenses such as Apache, MIT and BSD from 2009 to 2014.⁷⁷ Figure 1 below (reproduced from that study, which was based on Black Duck Software data) shows how companies’ license selection has changed over that timeframe. O’Grady notes that the GPL’s decline is related to the rise of permissive licenses:

[T]he largest single contributing factor to the decline of the GPL’s dominance – it’s worth reiterating, however, that it remains the most popular surveyed license – is the rise of permissive licenses. The two biggest gainers on the above chart, the Apache and MIT licenses, were collectively up 27%. With the BSD license up 1%, the three most popular permissive licenses are collectively up nearly 30% in the aggregate.⁷⁸

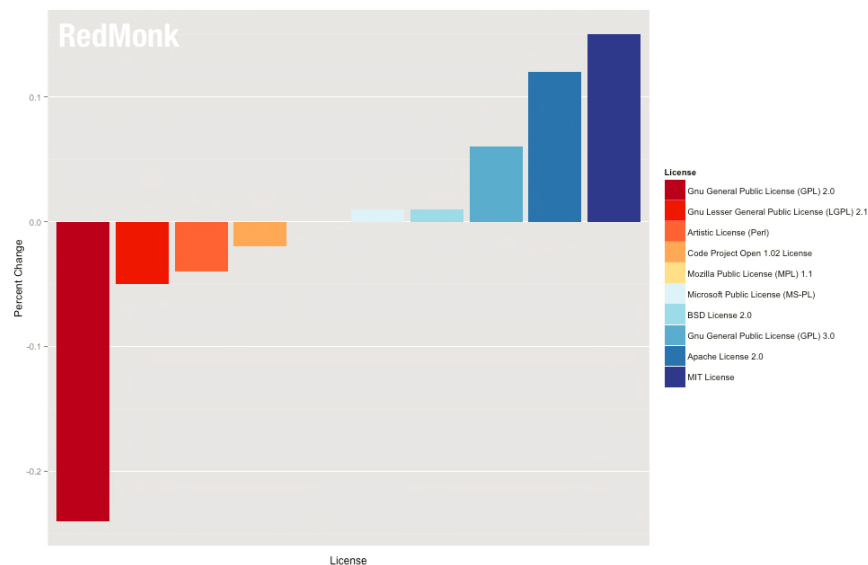


Figure 1: Change in Black Duck Open Source License Share: 2009-2014⁷⁹

⁷⁶ <http://www.marketwired.com/press-release/openlogic-scanning-data-reveals-oss-developers-choose-gpl-enterprises-prefer-apache-1514793.htm> (last visited Feb. 7, 2016) (emphasis added).

⁷⁷ <http://redmonk.com/sogrady/2014/11/14/open-source-licenses/> (last visited Feb. 7, 2016); <http://www.zdnet.com/article/the-fall-of-gpl-and-the-rise-of-permissive-open-source-licenses/> (last visited Feb. 7, 2016).

⁷⁸ <http://redmonk.com/sogrady/2014/11/14/open-source-licenses/> (last visited Feb. 7, 2016).

⁷⁹ *Id.* (Note that the Y-axis is displayed in raw numbers, not in percentage. So “-0.1” refers to a 10% decrease, etc.)

100. Those in the industry have commented that the concern about copyleft licenses such as the GPL is particularly prevalent among large commercial companies. For example, an analyst with 451 Research explains:

We typically see organizations focusing on some licenses that they approve for internal users or in products and licenses they do not support or condone. This is often the GPL, which is considered less permissive than other popular open source licenses, such as Apache Public License or Eclipse Public License.” The requirements of the GPL, compared to a more permissive license, “are still perceived and truly are in some cases more onerous to the user, especially if it’s a large enterprise that has high sensitivity around intellectual property, both others’ and its own.”⁸⁰

101. Similarly, Marcel de Vries, CTO of Xpirit (a technology consultancy) gave a talk on “Best Practices for Using Open Source Software in the Enterprise” in the TechEd Europe conference in 2014. He discussed the variety of open source licenses available. In this talk he explains why “GPL licenses scare enterprises a lot” as they might need to publish all their “cool stuff” and their intellectual property to the entire world.⁸¹

⁸⁰ <http://www.infoworld.com/article/2608340/open-source-software/why-gpl-still-gives-enterprises-the-jitters.html> (last visited Jan. 29, 2016).

⁸¹ <https://www.youtube.com/watch?v=HOCPD3v-Tx8> (last visited Jan. 29, 2016).

[15:16] “If you look at the different ways to look at the licensing spectrum; you can see that – well – on one hand we have the restrictive licenses and on the other hand we have the very permissive licenses - you can do more or less anything with it. And if you look at the red box over here [referring to the copyleft licenses, GPL, LGPL, AGPL] that is the stuff everybody’s talking about – and that’s what everybody is scared about and that’s what enterprises face something...‘we don’t go there.’ Because what this means is that we have this notion of copyleft.

Once you write code, this code is automatically copyrighted. Copyright. So the idea here is that copyleft is more like a joke, right? So instead of copyright, we’re doing copyleft. What does it mean? It’s all a sudden free? And I am just throwing away all my rights? Well the whole idea behind the copyleft is that you very specifically restrict what can be done with the source that you publish and with the GPL licenses, the LGPL and AGPL licenses, you’re more or less saying that once you use this software and you’re making changes to the software and then you start distributing the software - all of a sudden you need to distribute your own IP as well. It needs to be part of that, you need to have your software that built with that open source stuff under the same license as the stuff that you’ve been using.

So that’s the reason that these copyleft licenses – as they’re called in a more general way – in the general pocket, are called the viral licenses and that’s what everybody is scared of. So all of a sudden I’m writing this code, I have

102. This general trend away from GPL licenses, particularly in large enterprise businesses, further illustrates the risks and uncertainties that companies perceive to be associated with these licenses.

VII. Counseling Companies on FOSS Compliance

103. Whether or not a piece of FOSS software will *technically* operate in an intended or desired way is only the beginning of the inquiry into whether a company should incorporate particular FOSS software into its products and/or release its own software under a FOSS license. Each company's policy and management of FOSS depends heavily upon the culture and business model particular to that company. A company's decision to use FOSS (and if so which license to use) requires a highly fact and technology specific inquiry and should take into account multiple commercial, technical, business, competitive and other factors.

104. Just as with other corporate compliance programs, there is no one FOSS compliance solution that works for every company. For it to be successful, a company's FOSS program must be designed to reflect and accommodate the company's particular business plan and strategy, its product line and revenue model (i.e., whether it makes money through royalty-based licensing, a SaaS model, professional services, etc.), what its product(s) do, who its competitors are, the relative sophistication of its developer team and the company's level of risk aversion. Perhaps equally important are the company's culture and the way that engineers and lawyers interact. Some companies are highly process oriented, and the legal department has more "control" over what behaviors are and are not permitted. In those environments, it may be easier for the legal team to impose and implement policies and processes around FOSS. In other cases, where companies are built and thrive on chaos, counsel's attempt to impose a policy restricting use of FOSS is likely to flounder, and more dependence on developers' independent understanding of FOSS licensing considerations is warranted.

my intellectual property of doing some cool stuff and all of a sudden I find out that I am being using GPL license and now all of a sudden everything is out in the open and that's what scares a lot of enterprises because it might not be apparent straight away because just somebody pulls in some sources and you're doing your stuff – and then somebody comes by and say well I didn't expect your stuff there and you have my code in there and all of a sudden you're obligated to publish that to the entire world and not to even mention the lawsuits that you can get into and the money you need to pay for the infringement on the copyleft license in that particular case."

105. FOSS license compliance is not a process that should happen just at the moment of software intake. Rather, an effective compliance program will track the path of FOSS from its initial intake at the testing and evaluation stage through development, production and distribution, including any re-distribution or downstream distribution by resellers or Original Equipment Manufacturers, or “OEMs.” Companies that rely upon a closed-source model for marketing their software products need to be aware of how FOSS travels throughout their product development and production processes – and their FOSS management practices should include careful code audits and reviews throughout the software development and release process. On the other hand, given that FOSS license requirements are triggered upon distribution, companies whose business is SaaS-based may not need to be as vigilant – although they should not relax into thinking that they can bring in GPL’d or other “copyleft” software and simply use it for back-end server operation without monitoring their compliance.

106. When taking on new clients who ask me to advise them with respect to their FOSS license compliance, I generally recommend three actions at the outset:

- First, that we conduct an audit of the company’s existing code base, to identify any and all third party code contained in the codebase and determine what inbound licenses apply to that code. This audit may be conducted (i) manually, for example, by looking through the source code tree (start-up clients with limited budgets and smaller codebases often select this route), or (ii) electronically, via use of a scanning or other compliance tool such as Black Duck Software, Palamida, OpenLogic, or other similar tool (this latter approach is preferable for companies that have large code bases). In either case, I will carefully review the results of this audit with the client and work with them to ensure that we are taking appropriate action to be in compliance with all licenses governing the client’s use of third party code. During this review, I will ask the same kinds of questions that I ask with respect to FOSS intake, as discussed below.
- Next, or simultaneously with the step above, I ask to set up training sessions on FOSS licensing with both the client legal teams (if any) and product development and engineering teams. I believe that conducting

regular training sessions with the client development teams to provide an overview of FOSS licensing and compliance requirements, is critical for an effective FOSS compliance program. I frequently am surprised by how little development teams know about the ins and outs of FOSS licensing requirements, even when these teams include sophisticated developers that have been using FOSS software for years. For example, I find that many developers assume that LGPL-licensed code is “safe” to use and consider the LGPL equivalent to a permissive license. Needless to say, I try to educate them to think otherwise.

- Finally, and after consulting with the client about their company culture for the reasons discussed above, I work with the client to draft and create a FOSS compliance program that is appropriate for the company. This usually includes drafting a FOSS compliance policy that is distributed in particular to the engineering teams, as well as a FOSS intake questionnaire that includes questions about the FOSS code being used or being sought to be used such as those discussed below.

107. When working with clients on their proposed intake of any FOSS code and/or reviewing their use of FOSS as revealed by an audit, I go through the following process: First, I endeavor to have a good understanding of the client’s product(s), its business model, and its method(s) of distribution. I usually ask the client to provide me a diagram showing its product software architecture (and hardware architecture if the software is embedded in a hardware product). Then, with respect to each item of software that the client proposes to introduce – or has introduced – into its code base, I ask a variety of questions, depending upon the license governing the proposed inbound software, its proposed use, its functionality, where it fits in the software architecture, whether and how it will be distributed, and how critical it is to the functioning of the overall product. One reason for asking these and other questions is to be sure that we have identified any potentially copyleft licenses that may govern the client’s use of the inbound code, and to ensure that we take all steps necessary to comply with the copyleft terms of those licenses, or else that we substitute the proposed copyleft FOSS with a different solution that has license terms with which we can comply. This discussion may also contemplate the release in source code form of the client’s modifications to the inbound FOSS. Additionally, we want to be sure that we have identified correctly all FOSS code and licenses so that we can

provide the proper license and copyright notices along with our distribution of any FOSS or work based on that FOSS.

108. I ask the client developers to provide date of download, license version, and to confirm the website(s) from which they have downloaded or propose to download any FOSS. The latter request is in order to be able to confirm which particular license version governed the client's use of the software as of the date of download, and to encourage the client to track any potential license changes going forward. I also advise the client that, if it is not doing so already, it should maintain its code in a version control system such as CVS or Subversion so that it can monitor what code comes in to the source code tree, when, and under which license.

109. With respect to any proposed intake of FOSS code, I ask the client first to provide me with information about the FOSS code, including (by way of example):

- The name of the FOSS component, including version number
- The website from which the FOSS was or is to be downloaded (or other source, if not downloaded)
- The date of download
- The applicable license (including whether the license permits license migration over time; i.e., "GPLv3 or later version")
- The url indicating which FOSS license applies, and/or other locations indicating which license applies such as in the code itself or notice file accompanying the code

110. When the use of proposed inbound software is governed by a permissive license, I pursue a less extensive inquiry. That said, I want to be sure that the client understands that even permissive licenses do require proper copyright attribution and publication of the applicable license. In addition, there may be other requirements. For example, as noted above, the Apache License 2.0 contains explicit patent license and patent defense provisions that the client should consider prior to using and distributing Apache License 2.0 code. Further, the Apache license requires notice upon distribution if the client has made modifications to the Apache-licensed code.

111. If the proposed inbound code is licensed under a file-based copyleft license, or a full blown copyleft license, the inquiry delves deeper. A few of the many questions I will ask might include:

- Are licensor(s) individuals, a business, a non-profit FOSS project, academic institution, or other?

- Is the licensor part of a larger software project or otherwise reasonably well known in the industry? What is the licensor's reputation as a FOSS developer?
- How is the FOSS intended to be used in Client's products? And in which products?
- What is the date of first release of the Client product or service to Client customers?
- Is the Client product being licensed or sold via "downstream" OEMs or resellers?
- Is the FOSS available under any other license? For example, a commercial license or a different FOSS license? If so, why is the selected license preferable, or is it?
- Which of the following characterizes how the inbound software is or will be used?
 - Client internal tool (e.g., tools used for internal testing or in development, such as compilers, converters, debuggers or parsers).
 - Client internal use (software applications or services used by Client for internal purposes, such as intranet web server software).
 - Web service use (used as part of a service, including websites that resides on Client servers but is accessible outside of Client).
 - Customer/ Product Use (used in or with a Client product that is shipped to a Customer, installed at a Customer site or provided on media or via any other mechanism to Customer sites).
 - To create a platform on which client, customer or third party software will run?
- Has the Client modified any of the Software as part of the project or product and, if so, how?
- Is the Software being linked (statically or dynamically) to any other code?
- Is the Software and any other code run linked together in a shared address space?
- Is this Software providing core functionality for the Client product?
- Would the Client Product continue to function properly without it?
- What kind of time and effort would be involved if the Client were required to write code with similar functionality if it cannot comply with inbound FOSS license requirements?

112. Depending on the answers to questions such as those above, I may or may not request an in-person or web-based meeting with the applicable development team. Often, at such meetings, I will review the software architecture diagram with the development and/or product teams and endeavor to understand whether the client's proposed use leads to copyleft or other implications for the client. Based on that review, I will recommend, or not, that the client proceed with the proposed FOSS intake or usage.

113. As when counseling clients regarding third party software intake, when advising them on potentially out-licensing their home-grown code, a thorough understanding of their product and business model is key. Again, I ask a variety of questions, including their motivations for "open sourcing" their software, as in:

- What is the primary reason for wanting to release this code?
 - Promoting widespread adoption?

- Creating a widely adopted platform on which client, customer or third party software may be written and run?
- Wanting to let potential customers try out code before purchasing a commercial license?
- Creating an incentive for customers to buy a commercial license, under a dual licensing model (discussed below)?
- Wanting to leverage the diverse FOSS development community
- Added technical support and bug fixes?
- Visibility in the open source community?
- Being seen as a good player in the open source community?
- What is the competitive impact of releasing this code as open source?
- Is the client planning to release the code under an existing software project (e.g., Apache, Eclipse), or is the client planning to administer the release as its own project?
- If releasing as a new project, is there executive buy-in and commitment of resources to ensure that the project will be maintained and supported over time?
- Does the client have the rights to release the software as proposed?
 - What third party software, if any, is included in the software proposed to be released?
 - Is any license governing software included in the code incompatible with the proposed outbound license?

114. Given that copyleft open source licenses may require release of a company's proprietary source code if the program is distributed, one might expect that the for-profit companies would "run away screaming" from using copyleft FOSS or releasing their own code as FOSS. This often is not the case. In fact, numerous companies are using copyleft licenses precisely *in order to* generate revenue. For example, companies may choose to release their own code under a copyleft license in order to:

- Encourage widespread adoption of their software product so that they can then generate revenue from other technical support and/or consulting services relating to that software.
- More rapidly build and develop their own software, by releasing their code as open source and receiving back bug fixes and enhanced features from the FOSS developer community.
- Build their reputation in the FOSS Community as a good citizen and team player in that Community.
- Encourage sale of their own hardware products, by making the software that runs on those hardware products publicly and widely available.
- Make their software product available for testing, evaluation and/or open source development, while offering a commercial license that licensees can buy and that does not contain the same legal obligations as the copyleft license.⁸²

⁸² Mr. Hall discussed such a model in a recent talk titled "Open Source Licensing and Business Models" at All Things Open 2014:

- Release their software product with a limited feature set as a “community edition,” but offer an enhanced or “enterprise” edition for a fee under a commercial license. These last two models are known as “dual licensing” models and are commonly used by companies that are building a business around FOSS.

VIII. Background on Java

115. Java is a programming platform created in 1995 by Sun Microsystems, Inc (“Sun”).⁸³ Sun was then acquired by Oracle Corporation in 2010 and became Oracle America, Inc. (collectively, “Oracle”).⁸⁴ I understand that the Java platform is an applications platform and includes a set of technology resources to build and run programs. I further understand the platform consists of the Java programming language, the Java Runtime Environment, which features the Java VM (Virtual Machine) and the Java API (Application Program Interfaces) packages.

116. Java was designed in accordance with the “Write Once, Run Anywhere” (WORA) philosophy for cross platform use. That is to say, once a Java program is developed on a device for a specific edition of Java, it can be assured to work in the same way on all devices that support that edition as well.⁸⁵ This portability enables the Java platform to be used by a large and active community of software developers and programmers.

Dual licensing, which most of you are probably familiar with, is the idea of offering the same code under both a commercial and an open source license. And when it’s dual licensed by a commercial company, they will almost always choose what’s thought of as a closed source unfriendly or commercially unfriendly license. So that’s almost always going to be GPL or AGPL or something that, for a commercial company that wants to modify it and combine it with their software, they don’t want to make their source code available. And so, to avoid those effects that the companies don’t want to deal with, they can take the commercial license and not have to comply. It’s a way to kind of drive up adoption, get your product out there. That’s the best thing. For open source licensing for companies, it’s really all about adoption. That’s the best thing that’ll do, is drive adoption.

available at https://www.youtube.com/watch?v=OhQD6WD_KYE.

⁸³ <http://www.freejavaguide.com/history.html> (last visited Feb. 7, 2016).

⁸⁴ <http://www.oracle.com/us/corporate/press/018363> (last visited Feb. 7, 2016) (acquisition was originally announced in 2009).

⁸⁵ <http://www.computerweekly.com/feature/Write-once-run-anywhere> (last visited Feb. 5, 2016).

117. I understand from Dr. Schmidt's Opening report that the Java API packages serve as a set of prewritten programs that can be used by software developers in creating their own software programs for the Java platform. I understand that the following 37 Java API packages are at issue:

java.awt.font	java.nio.channels.spi	java.util	javax.net.ssl
java.beans	java.nio.charset	java.util.jar	javax.security.auth
java.io	java.nio.charset.spi	java.util.logging	javax.security.auth.callback
java.lang	java.security	java.util.prefs	javax.security.auth.login
java.lang.annotation	java.security.acl	java.util.regex	javax.security.auth.x500
java.lang.ref	java.security.cert	java.util.zip	javax.security.cert
java.lang.reflect	java.security.interfaces	javax.crypto	javax.crypto.interfaces
java.net	java.security.spec	javax.sql	
java.nio	java.sql	javax.crypto.spec	
java.nio.channels	java.text	javax.net	

118. In 2007, Sun announced that it would release open source versions of the Java platform, known as "OpenJDK."⁸⁶ It announced that the Java API libraries (including the 37 Java API packages at issue here), would be licensed under GPLv2-CE.⁸⁷

119. Through its acquisition of Sun, Oracle is now the owner and distributor of the Java platform and software products. Since 2010, two new editions of the platform have been released: Java SE 7 in 2011⁸⁸ and Java SE 8, the current edition released in March 2014.⁸⁹

IX. Background on Android

120. Android began as Google's mobile applications platform, which was first announced in 2007 and then released in 2008.⁹⁰ Since its release, adoption has been rapid and Android had 1.4 billion active users as of 2015.⁹¹ Today, Android is used for mobile devices,

⁸⁶ <https://web.archive.org/web/20070510070408/http://www.sun.com/aboutsun/pr/2007-05/sunflash.20070508.3.xml> (last visited Feb. 5, 2016).

⁸⁷ <http://openjdk.java.net/faq/> (last visited Feb. 7, 2016).

⁸⁸ http://www.oracle.com/webapps/events/ns/EventsDetail.jsp?p_eventId=134208&src=7299693&src=7299693&Act=18 (last visited Feb. 5, 2016).

⁸⁹ <http://www.oracle.com/us/corporate/press/2172618/> (last visited Feb. 5, 2016).

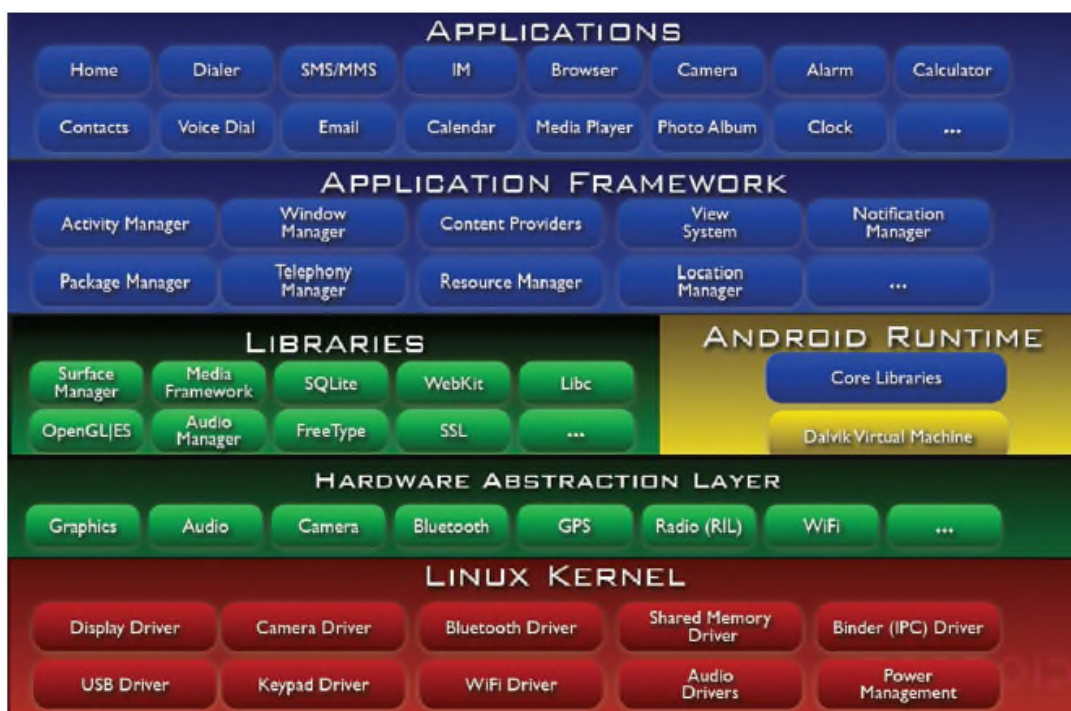
⁹⁰ <http://android-developers.blogspot.in/2008/09/announcing-android-10-sdk-release-1.html> (last visited Feb. 5, 2016).

⁹¹ <http://techcrunch.com/2015/09/29/android-now-has-1-4bn-30-day-active-devices-globally/> (last visited Feb. 5, 2016).

including smartphones and tablets, and is expanding into markets for TVs/set-top boxes, wearables, auto, the Internet of Things, etc.⁹²

121. When Google announced Android in November 2007 it also announced the formation of the Open Handset Alliance (“OHA”), a consortium that began with 34 member organizations and has since expanded its membership to 84.⁹³ The mission statement of the alliance is to “accelerate innovation in mobile and offering consumers a richer, less expensive, and better mobile experience.”⁹⁴ Members of OHA include mobile operators such as Sprint, Nextel, and T-Mobile, handset manufacturers such as Acer, LG, and HTC, semiconductor Companies such as Intel and NVIDIA, software companies such as Ebay, as well as commercialization companies such as Accenture and Noser.⁹⁵

122. I understand that the architecture of the Android stack is illustrated in the following figure:



⁹² <http://www.cbsnews.com/news/google-io-android-tv-smart-homes-wearables-and-more/> (last visited Feb. 8, 2016).

⁹³ http://www.openhandsetalliance.com/press_110507.html (last visited Feb. 8, 2016).

⁹⁴ http://www.openhandsetalliance.com/oha_faqs.html (last visited Feb. 5, 2016).

⁹⁵ http://www.openhandsetalliance.com/oha_members.html (last visited Feb. 7, 2016).

123. I further understand from Dr. Schmidt that Google makes its source code for most components of the Android stack available for download (for, e.g., developers and OEMs) over the Internet.

X. Google's Statements Regarding Potential Inclusion of Copyleft Licenses in Android

124. I understand from Mr. Hall and Dr. Astrachan's reports that they believe Google could have adopted an OpenJDK implementation of the 37 Java API packages in 2007.⁹⁶ In reaching these conclusions, they do not touch on the business realities and legal risks associated with Google's use of OpenJDK code in Android. In particular, Mr. Hall's report does not seem to consider, let alone analyze, the business and risk factors that Android and its OEMs would have considered—and in fact did consider—when choosing whether to incorporate OpenJDK class libraries into Android. Because Mr. Hall and Dr. Astrachan do not appear to have analyzed the legal or business risks associated with the use of the GPLv2.-CE-licensed OpenJDK code, their conclusion that Google could have used OpenJDK in 2007 is necessarily flawed.

125. The website for the Open Handset Alliance (“OHA”) publishes a FAQ explaining that the reason for choosing the Apache License (also referred to as “Apache v2”) for Android is to enable handset manufacturers to keep their innovations and differentiated features as closed source, which could not have been done by using other licenses.⁹⁷

Why did you pick the Apache v2 open source license?

Apache is a commercial-friendly [sic] open source license. The Apache license allows manufacturers and mobile operators to innovate using the platform without the requirement to contribute those innovations back to the open source community. Because these innovations and differentiated features can be kept proprietary, manufacturers and mobile operators are protected from the “viral infection” problem often associated with other licenses.

126. On the day of the Android launch, a conference call was held and recorded with participants from Google, T-Mobile, HTC, Qualcomm and Motorola to discuss the Android platform. In this conference call, Andy Rubin (co-founder of Android) explained

⁹⁶ Hall Report, ¶122; Astrachan Report, ¶264.

⁹⁷ http://www.openhandsetalliance.com/android_faq.html (last visited Feb. 1, 2016) (Note this FAQ was added in November 2007 so it was contemporaneous with the actual decision of which license to select (i.e., it was not written with the benefit of hindsight).

Google's rationale for releasing Android under the Apache License, consistent with what was posted on the FAQ:

ROB PEGORARO, MEDIA, WASHINGTON POST: Thanks. My question here, in reading this FAQ online, it sounds--and correct me if I'm wrong--as if the software will ship with a certain set of capabilities. Carriers can modify those to remove things they don't want. Is there any minimum set of requirements that carriers have to include, or I guess handset manufacturers as well, for it to be an Android-powered phone?

ANDY RUBIN: This is Andy. I'll take that answer. No. The purpose of it being open and providing it as open source, I mean, anybody can do whatever they want with it. But the particular license we chose which is the Apache V2 license, has no restrictions.⁹⁸

127. In a press conference discussing the launch of Android, Google CEO Eric Schmidt highlighted the benefits of permissive Apache licensing for ecosystem partners:

This will be the first fully-integrated software stack, including an operating system and middleware, being made available under the most liberal open-source license ever given to mobile operators [and handset makers].⁹⁹

128. Android co-founder Rich Miner also acknowledged in 2007 the advantage of permitting companies to differentiate on the Android platform without releasing their own intellectual property and that the use of the Apache license would serve this goal.

The open model will also allow entrepreneurs to reach a large consumer base without sacrificing their ownership rights. The Apache license we are using will enable people with strong enterprise software to bring their applications to the Android platform without open-sourcing their intellectual property.¹⁰⁰

129. In 2008, Andy Rubin confirmed that Google deliberately chose to release Android under the Apache license in order to avoid the risks that copyleft licenses might impose on OEMs. In an interview to CNET in 2008, he explained:

The thing that worries me about GPL is this: suppose Samsung wants to build a phone that's different in features and functionalities than (one from)

⁹⁸ FD Wire, Google, T-Mobile, HTC, Qualcomm and Motorola to Discuss New Open Platform for Mobile Devices – Final, November 5, 2007.

⁹⁹ <http://techcrunch.com/2007/11/05/breaking-google-announces-android-and-open-handset-alliance/> (last visited Feb. 1, 2016).

¹⁰⁰ “Google goes mobile with Android plan”, Neon Kelly, Computing, Incisive Media November 15, 2007.

LG. If everything on the phone was GPL, any applications or user interface enhancements that Samsung did, they would have to contribute back. At the application layer, GPL doesn't work.¹⁰¹

130. Thus, a year after Sun announced the release of OpenJDK under the GPLv2-CE license, Andy Rubin publicly acknowledged that the GPL licenses posed risk to OEMs and concluded that such a license “doesn't work” for Android.

131. In other public statements during the same time frame, Google apparently continued to express a concern over what using the GPLv2-CE license for Android code would mean to its OEMs. For example, in a report on the 2008 Google I/O conference, one observer explained:

The first and main reason they give us for using Harmony instead of OpenJDK is the GNU license (GPL). Cell phone makers want to link proprietary value-add code directly into the system (into JVM-based apps. and/or service processes), and they do not want to worry about copyleft. Perhaps there is some education needed here about the classpath exception. (I know I don't understand it; maybe they don't either. *And* their license works appear to have a well-considered preference for Apache 2 over GPL+CPE.)¹⁰²

132. Android's use of the permissive Apache license apparently was a selling point for its technology partners and OEMs. In 2008, Texas Instruments announced that its Bluetooth and Wireless LAN software drivers were available for the Android platform in a press release. In that press release, Texas Instruments mentioned the importance of the Apache license and pointed out that permissive licenses such as Apache and BSD enable companies to differentiate on the Android platform without releasing their own intellectual property:

Pre-integrating the codes into the Android platform gives developers and handset manufacturers a comprehensive set of tools that simplify the innovation process and reduce time to market. Additionally, TI leverages the Apache and BSD-style licenses in support of the Android platform. With these licenses, **a broader number of companies will be able to adopt the Android platform and build on top of it without having to expose proprietary technologies.** The flexibility offered with these licenses

¹⁰¹ <http://www.cnet.com/news/why-oracle-not-sun-sued-google-over-java/> (last visited Jan. 30, 2016).

¹⁰² https://blogs.oracle.com/jrose/entry/with_android_and_dalvik_at (last visited Jan. 29, 2016).

simplifies the development of consumer devices and encourages further adoption of open source software.¹⁰³

133. Similarly, a report issued by ABI Research (a technology market intelligence company) in December 2007 explained why Google's Android was a smart investment in platform development:

Android provides a ready-made ecosystem of vendors, carriers, and software developers that could provide enormous economy of scale right from the start," says Stuart Carlaw, ABI Research director. "Another concern over Linux involves how to monetize innovation that is subject to a public license. Android solves this through the use of the Apache V2 public license, which does not include a copyleft function."¹⁰⁴

134. At the 2008 Google I/O conference, Jason Chen, a Google Developer Advocate also explained how releasing Android under the Apache License would enable handset manufacturers and operators to customize the software and differentiate their products:

(Min 13:00)

"In mobile there are three groups of actors that are important. The first is the mobile industry. When I say industry, I am talking about handset manufacturers, operators, commercialization vendors, anyone who is really involved in bringing a device to market. And when we say that Android is open for the mobile industry it means a couple things. First, and we've talked a lot about this, the software stack will be open sourced under the Apache 2.0 license and that source will be available after the first handsets ship. Now some of what we're doing with Android is not... we can't license it under Apache, so things like our work in the Linux Kernel or Eclipse plugin, some of the other work that's GPL or licensed under other licenses and we've released that code already. **But everything else in terms of the runtime, the framework, all the libraries that I showed in that system stack, all those things will be open sourced after or will be available after the first handsets ship in the second half of this year.** And the last point to really make is that once it's open sourced, anyone can take that code and build a system image.

(Min: 35:00)

A: After the first devices are available on the market in the second half of this year, all of the code for Android will be available under the Apache 2 license, and that you can take as anyone could take and build as a system

¹⁰³ <http://newscenter.ti.com/index.php?s=32851&item=126078> (Oct. 28, 2008) (last visited Feb. 1, 2016) (emphasis added).

¹⁰⁴ *REPORT - GOOGLE'S ANDROID MAY JUSTIFY VAST INVESTMENT IN PLATFORM DEVELOPMENT* WirelessEurope's Daily Bulletin Copyright December 3, 2007.

image, install it on hardware, customize it to what you need - and if you want to build handsets, it's possible to do so. So it's not restricted.¹⁰⁵

135. The concerns relating to GPL'd code are also reflected in internal Google communications (not just in public statements from Google). For example, in an email to Bob Lee (a Google engineer), Andy Rubin discusses the challenges of using GPL in systems such as Android,¹⁰⁶ specifically highlighting the challenges it would create for OEMs and Carriers:

... and as far as GPL-ing the VM, everything that is linked with the VM would get infected.

The problem with GPL in embedded systems is that it's viral, and there is no way (for example) OEMs or Carriers to differentiate by adding proprietary works. We are building a platform where the entire purpose is to let people differentiate on top of it.

Finally, Sun has a different license for its library for SE and ME. The SE library is LGPL, ME library is GPL. That means anything that links with the ME library gets infected. And the SE library is not optimized for embedded systems.

Sun chose GPL for this exact reason so that companies would need to come back to them and take a direct license and pay royalties.

Tricky, no? Why would we want to do anything to support this behavior? We want to distance ourselves as much as possible from Sun.
-andy

136. Based on my reading of these statements from Google and other industry players made around the time that Android was first announced—the same time period in which Mr. Hall now contends Google could have used OpenJDK code licensed under GPLv2-CE—Google apparently recognized the copyleft effects of GPL licenses, as well as their associated risks and chose the Apache license precisely to avoid those effects and risks. This choice seems in part because Google understood that its OEMs would want to compete with one another by differentiating Android without any risk that those differentiations would be made public (and thus destroy the very basis for that competition). These statements suggest to me that it was unlikely that Google would have incorporated into the Android stack any GPL licensed code, including GPLv2-CE code, at the time—as Mr. Hall concludes it “could have” done. Had I been advising Google at the

¹⁰⁵ https://www.youtube.com/watch?v=x1ZZ-R3p_w8 (last visited Jan. 28, 2016).

¹⁰⁶ TX230.

time under those circumstances, I doubt that I would have advised it to use or rely upon any form of GPL'd code, as the legal uncertainty outlined above could have posed a substantial business risk to Google.

137. Another factor indicating that Google was concerned about the effect of copyleft licenses in the Android stack is that Google appears to have taken steps to try to avoid the copyleft implications of such licenses when including GPL'd code in Android. Google itself acknowledged its efforts to insulate GPL code from other parts of the Android stack. For example, during 2008 Google I/O Conference, Patrick Brady—a technology program manager at Google—gave a talk on the “Anatomy and Physiology of an Android.” When explaining that the Linux Kernel and kernel drivers in Android are subject to the GPL, Brady explained Google had to find a way to distribute Android under a different license because it believed that handset manufacturers and chip set providers would like to keep their intellectual property closed source.

“Now, any Linux gurus out there or people who are paying attention when I said earlier that the Linux Kernel provides a great abstraction layer probably asking well why do we have another abstraction layer on top of the Kernel? Well, the real reason is – and we debated this internally for a while – not all components have a standardized interface in the Kernel today. So you have things like a vibrator driver for your device, or an LED driver. These aren't always exposed as standard kernel drivers. And the other problem is kernel drivers are GPL, which exposes the intellectual property. **Like I said, that copyleft means that if you put anything in the Linux Kernel, it pretty much needs to be shared as open source software. And we found that as much as we want to ship everything as open source, there's a lot of handset manufacturers and chip set providers who feel they have a great deal of intellectual property in their drivers, so they want to keep those closed source. So we needed to provide a way to do that in user space outside of the Linux Kernel.**”¹⁰⁷

138. Brady further explained how having a Hardware Abstraction Layer, or “HAL,” was necessary to provide a buffer from the underlying Linux kernel so that Android would enable OEMs and handset manufacturers to keep their custom code proprietary “for things like audio and camera or Bluetooth, GPS, the RIL or radio interface layer to talk to the modem, baseband, wifi, and many more” as well as to differentiate and control “displays, audio and their touchpads.”

¹⁰⁷ <https://www.youtube.com/watch?v=G-36noTCaiA> (last visited Feb. 5, 2016).

So how do we sit all of this on top of these different Linux implementations? Many of the OEMs now have handset manufacturers, now have Linux distributions maybe on their mobile platforms, and they have the drivers to control their displays, and they have the drivers to control audio and their touchpads and things like that. And so we have a set of hardware abstraction libraries that really provide a better abstraction layer between the hardware and the upper layers of the Android platform. To give you just an idea of where these would sit and kind of changing the overall architecture diagram a little bit to show you that these are native libraries sitting down there in the libraries layer, and they contain abstractions for things like audio and camera or Bluetooth, GPS, the RIL or radio interface layer to talk to the modem, baseband, wifi, and many more.¹⁰⁸

139. Brady also explained that Google developed its own custom library (Bionic) instead of using a GPL-licensed version of the same (libc) to avoid the copyleft effect of GPL:

“Why Bionic? There is a lot of overhead involved in maintaining your own c library... We got a lot of questions from Linux Community about that... number one [reason], we wanted to address the licensing problem with the standard C library, standard C library is GPL - which has a *nasty* copyleft clause, which means anything that is built with GPL code needs to be open sourced itself, we have a lot of people who wanted to write proprietary code that contains intellectual property and keep that as closed source.”¹⁰⁹

140. As noted above, these statements cause me to doubt that Google would have incorporated into the Android Runtime any GPL license, including GPLv2-CE, at the time, and to question the relevance of Mr. Hall’s and Dr. Astrachan’s conclusions that Google technically “could have” done so.

XI. Google’s Announcement to Use OpenJDK

141. I understand that on December 24, 2015, Google published source code including OpenJDK-based code for the 37 API packages at issue in this case.¹¹⁰ I understand that the OpenJDK-based code for the 37 API packages at issue reside in a folder in the master branch of libcore named “ojluni.” I understand that Google has indicated it plans to release this OpenJDK-based code in a future version of Android.¹¹¹

¹⁰⁸ *Id.* (emphasis added).

¹⁰⁹ *Id.* (emphasis added).

¹¹⁰ <http://venturebeat.com/2015/12/29/google-confirms-next-android-version-wont-use-oracles-proprietary-java-apis/> (referencing a December 24 code commit) (last visited Feb. 2, 2016).

¹¹¹ *Id.*

142. As I have described earlier in this report, analysis of whether a particular implementation of GPLv2-CE-licensed code is compliant requires a fact and technology-specific analysis. Therefore, I cannot state unequivocally that Google's anticipated inclusion of OpenJDK-based code into its core libraries would or would not conform to the specific terms of the GPLv2-CE. Until the Android stack is finalized and released as an actual version of Android, it will not be possible able to form a definitive opinion here. However, based on my analysis of the GPLv2-CE, and my understanding of how Google has implemented the 37 class libraries at issue to date, I believe that Google could significantly risk exposing other portions of the Android stack to the copyleft provisions of the GPLv2 by using the OpenJDK code in its core libraries.

XII. Anatomy of GPLv2-CE

143. In his report, Mr. Hall concludes that Android's use of OpenJDK-based code in the 37 Java API packages would not subject other parts of the Android stack to the copyleft requirements of the GPLv2, GPLv2-CE or any other copyleft license.¹¹² However, Mr. Hall reaches this conclusion without offering any analysis of the specific provisions of the GPLv2-CE as they apply to Google's implementation of the OpenJDK-based code in Android. Before analyzing Android's implementation under the GPLv2-CE, it is important to first review the terms of the GPL itself, as these form the basis of the Classpath Exception.

(i) Preamble¹¹³

144. In relevant part, the Preamble of GPLv2 states:

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to **certain responsibilities for you if you distribute copies of the software, or if you modify it.**

¹¹² A. Hall's Opening Report at ¶¶118-120.

¹¹³ All quotes to the GPLv2-CE in this section come from <http://openjdk.java.net/legal/gplv2+ce.html>.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

145. As explained by this language, GPLv2's license obligations are triggered upon the *distribution* of the licensed Program (or modifications thereof). Provided those obligations are met, the license permits copying, distributing, and/or modifying the software. Running (using) a program without distributing it is expressly permitted without restriction.

(ii) Section 0

146. Section 0 defines key terms that are used throughout the license.

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. . . .

147. Thus, GPLv2 defines the Program as any work that is identified by the copyright holder as subject to the license. A "work based on the Program" is defined as "the Program or any derivative work under copyright law" including any "work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language."

148. Section 0 also defines the overall scope of the license and its application. It states that "[a]ctivities other than copying, distribution and modification are not covered by this License; they are outside its scope." In other words, the acts of copying, distributing, and modifying the licensed software are governed by the license. Any other actions, such as merely running with Program (without copying, modifying, or distributing all or any portion of the Program itself) do not trigger the requirements of GPLv2.

(ii) Section 1

149. Section 1 addresses rights to distribute verbatim copies of the Program:

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty...

This right to distribute unmodified code is limited to *source* code and explains that licensees have the right to distribute *verbatim* source code as long as that code is accompanied by the appropriate notices (including a copyright notice).

(iii) Section 2

150. Section 2 is where the copyleft aspects of the GPL begin to kick in, as it addresses licensees' rights to distribute modifications to the Program. This is a critical section of the license and is reproduced in full below:

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) *You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.*

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of *derivative or collective works based on the Program*.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a

storage or distribution medium does not bring the other work under the scope of this License.¹¹⁴

151. Section 2 first explains that licensees are allowed to modify the Program and distribute copies of their modifications. However, in doing so, they must include the same notices required by Section 1 and also meet the three additional requirements enumerated in subparagraphs (a)-(c).

152. Subsection (b) states that if a licensee modifies the Program and then distributes a work that contains or is derived from the Program, that *entire distributed work* must be distributed under the terms of GPLv2.

153. While Section 2 goes on to say that “[t]hese requirements apply to the modified work as a whole,” it carves out limited exceptions to the requirement to distribute the whole work under GPLv2. More specifically, if “identifiable sections” of the work “are not derived from the Program, and can be reasonably considered independent and separate works in themselves,” then the GPL terms do not apply to those discrete and independent portions if they are distributed separately from the rest of the work. However, the license is clear that if the same sections are distributed as “part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License.” In other words, if works that are not derived specifically from the GPL-licensed code are distributed with the GPL-licensed code as a “whole,” the GPL applies to those independent works.

154. Finally, the last paragraph of Section 2 includes a “mere aggregation clause” that permits distribution of that whole work with *a different* independent work on a storage or distribution medium without the GPLv2 obligations applying to that independent work.

(iv) Section 3

155. GPLv2 Section 3 addresses rights to copy and distribute the Program (or a work based on it as described in Section 2) in object code or executable form. This requires that, in addition to meeting the requirements of Sections 1 and 2, licensees must make the Program’s source code available when distributing the Program in object code or executable form.

¹¹⁴ Emphasis added.

(v) Section 4

156. Section 4 describes the ramifications if licensees do not comply with the terms of the GPLv2: in such case, the licensees' rights to use, copy, modify and distribute code under the terms of the license are automatically terminated.

(vi) Classpath Exception

157. Having parsed through the central terms of the GPL, one can then turn to analyzing the GPLv2 with Classpath Exception. The Classpath Exception is a "special exception" to the GPLv2 that describes a very specific carve-out to the Section 3 requirement that distribution of the Program in executable form be made subject to the terms of the GPL. It reads as follows:

Linking this library statically or dynamically with other modules is making a combined work based on this library. Thus, the terms and conditions of the GNU General Public License cover the whole combination.

As a special exception, the copyright holders of this library give you *permission to link this library with independent modules to produce an executable*, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. *An independent module is a module which is not derived from or based on this library.* If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.¹¹⁵

158. The Classpath Exception first confirms the provisions of Sections 2 and 3 of GPLv2, when it states that "[l]inking this library statically or dynamically with other modules *is making a combined work based on this library.*" It then affords a special exception in one specific circumstance: when a licensee links the "library with independent modules to produce an executable," it is permitted to distribute that *executable* under the terms of its choice (i.e., not under the GPLv2-CE). Note that the exception does *not* apply to the distribution of *source code* that is covered by GPLv2; in other words, distributions of source code would not be covered by the exception and thus would be subject to the terms of GPLv2.

¹¹⁵ Emphasis added.

159. I understand from Dr. Schmidt that the term “executable” is a well-known concept in computer programming; it is a file capable of being launched to perform a task in accordance with designated instructions (e.g., an ‘.exe’ file that can be launched on a Windows computer). Importantly, an executable is distinctly different from executable code, and both are different from source code itself.

160. In his report, Mr. Hall concludes that “the GPL-2.0-CE license exempts those qualifying for the Classpath Exception from *any* compliance obligations related to the GPL-2.0-CE-licensed software.”¹¹⁶ I disagree with his conclusion. There is nothing in the Classpath Exception that provides a broad exemption from *all* of the other requirements of GPLv2. The Classpath Exception does not say anything about eliminating the underlying GPL requirements to distribute source code of the Program itself (including works based upon the Program) as required under Sections 2 and 3 – in fact, it confirms that linking the “library statically or dynamically with other modules is making a combined work based on [the] library.” There is nothing stated in the Classpath Exception, nor does Mr. Hall point to any language in its terms, that would permit all of the other requirements of GPLv2 to be discarded, as Mr. Hall suggests. Rather, the Classpath Exception permits redistribution of an *executable* only, and then only when that executable results from the linking of the licensed library to an independent module.

XIII. Google’s Proposed Use of OpenJDK Poses Risks to Android and its OEMs

161. As previously noted, on December 24, 2015, Google published source code in its master branch using OpenJDK-based source code for the 37 API packages at issue in this case.¹¹⁷ That OpenJDK-based source code is purported to be licensed under GPLv2-CE, while Google continues to distribute the majority of the rest of the Android stack under the Apache License, v2.0.¹¹⁸

162. Mr. Hall appears to conclude that the GPLv2-CE permits licensees qualifying for the Classpath Exception to ignore all other provisions of the license. He states: “[T]he GPL-

¹¹⁶ A. Hall’s Opening Report, at ¶61 (emphasis added).

¹¹⁷ <http://venturebeat.com/2015/12/29/google-confirms-next-android-version-wont-use-oracles-proprietary-java-apis/> (last visited Feb. 2, 2016).

¹¹⁸ See, e.g., https://android.googlesource.com/platform/libcore/+/_master/luni/src/main/java/java/net/AddressCache.java (last visited Feb. 5, 2016) (showing an Apache license in the header of another file in the master libcore branch).

2.0-CE license exempts those qualifying for the Classpath Exception from *any* compliance obligations related to the GPL-2.0-CE-licensed software.” (emphasis added). He goes on to say that “the only obligation placed on distributors qualifying for the Classpath Exception is compliance with the requirements imposed by whichever other license(s) may be covering the *linking* GPL-incompatible software.” (emphasis in original). Again, I disagree. As discussed above, if a licensee modifies the GPLv2-CE-licensed code and then distributes that modified code in source code form, it must meet the notice and distribution requirements of Section 2. Further, if the licensee distributes its *modified* GPLv2-CE-licensed code in object code or executable form, it must make the source code to that modified code available as well. It is *only* if the licensee links the GPLv2-CE licensed code to an independent module and thus creates and distributes the “resulting executable” that the Classpath Exception would come into play.

163. In the case of the distribution of Android, and with respect to the 37 Java API packages at issue, it is my understanding from Dr. Schmidt that Google has not merely “linked” these libraries to create an executable that it then distributes. Rather, my understanding is that it has distributed the libraries as part of the Android stack, and in source code form. If my understanding is correct, the Classpath Exception would not apply to Google’s distribution of the 37 API packages it copied from OpenJDK 8.

164. Mr. Hall states that the copyleft implications of GPLv2 would apply only to modifications to the OpenJDK-based source code itself, “but not to software linking with those packages.”¹¹⁹ But in reaching this conclusion, Mr. Hall’s report does not analyze how Google actually generates and distributes the Android stack, much less how Android OEMs do so. Taking into account all relevant facts and license language, it is my opinion that Google’s incorporation of OpenJDK-based code in the master branch of Android poses a significant risk that additional elements of the Android stack (including modifications made by OEMs) would be subject to the requirements of GPLv2-CE.

165. It is interesting that Mr. Hall’s own characterization of the Android Runtime, which includes all of Android’s core libraries (not just the 37 Java API packages) and Android’s virtual machine, implies that the entire Android Runtime is subject to GPLv2. Mr. Hall states in paragraph 116 of his report that “On Dec. 24, 2015, Google publicly released source code for a modified version of the Android Runtime. That version is **derived from**

¹¹⁹ A. Hall’s Opening Report, at ¶118.

37 Java API packages from the OpenJDK project that are licensed under the GPL-2.0-CE license in place of the 37 API packages previously derived from the Apache Harmony project that were licensed under the Apache-2.0 license.” (emphasis added). As discussed above, Section 2 of GPLv2 requires distribution of any work that is *based on or derived from* the Program. If Mr. Hall’s characterization of Android Runtime as *derived from* the OpenJDK-based code is accurate, all of Android Runtime would need to be released under GPLv2-CE in order to comply with copyleft provisions of that license. Recall that the Classpath Exception only applies to “independent modules,” which are defined as “not derived from or based on this library.” So, if Mr. Hall is correct that the Android Runtime “is derived from [the] 37 Java API packages from the OpenJDK project,” Android Runtime necessarily would need to be released under GPLv2-CE (or GPLv2).

A. Google’s Proposed Use of OpenJDK Code Could Expose Parts of the Android Stack to GPLv2 Obligations

166. I understand from Dr. Schmidt’s report that Google has copied from and modified the OpenJDK-based libraries that it now publishes in source code form.¹²⁰ I understand further that there are other elements of the Android stack that are distributed collectively, in source code form and as a complete work, with the OpenJDK-based libraries to OEMs.¹²¹ I understand that many of these elements are specifically designed to directly integrate with the modified OpenJDK libraries.¹²² Google publishes the OpenJDK-based libraries with a notice stating that they are subject to the GPLv2-CE license, and therefore meet the definition of the “Program” in that license.¹²³

167. It is my opinion that there is a significant risk that GPLv2-CE requires source code of at least some elements of the Android stack—in addition to the OpenJDK-based code—to be distributed under the GPLv2-CE license, which Google has not done.¹²⁴

¹²⁰ Expert report of Professor Doug Schmidt, February 8th, 2016.

¹²¹ *Id.*

¹²² *Id.*

¹²³ See, e.g., https://android.googlesource.com/platform/libcore/+/_master/ojluni/src/main/java/java/security/AccessControlContext.java (last visited Feb. 5, 2016).

¹²⁴ As far as I am aware, the only source code that Google has published in the Android stack under GPLv2-CE is the ojluni folder containing the 37 Java API packages based on OpenJDK code.

168. In my opinion, if Google modifies the Program (i.e. the OpenJDK libraries) and then distributes the modifications, Section 2 is triggered. Mr. Hall appears to agree with this proposition, as he states that “GPL-2.0-CE imposes a copyleft effect on distributed *modifications* to the OpenJDK-based Java API packages.”¹²⁵ (emphasis in original). However, Mr. Hall does not address the additional requirements of Section 2—that any source code distribution of any *work* containing the Program must be licensed as a whole under GPLv2. Nor does he address the linking of the Program with modules that are not independent modules. Nor does he address the distribution of the Program “as part of a whole which is a work based on the Program,” as is contemplated in Section 2. Considering the totality of Section 2, Google’s inclusion of GPLv2-CE code in the Android stack poses a significant risk that additional aspects of the Android stack are subject to the requirements of GPLv2.

169. It is my understanding from Dr. Schmidt’s report that Google distributes most elements of Android as a whole.¹²⁶ I further understand that three examples of elements from the Android stack (as shown in the Android stack diagram I reference above) that are distributed with the modified Program (OpenJDK-based libraries) are:

- Application Framework - elements of this framework are used within the complete work and directly use the modified OpenJDK libraries;
- Applications - applications are distributed with the complete work and directly use the modified OpenJDK libraries; and
- Bionic library – the modified OpenJDK library directly interacts with this library through JNI calls.

170. I understand that source code for these three pieces of the Android stack are distributed together with the modified OpenJDK source code for the 37 API packages at issue and are designed to function together as a part of a whole work.¹²⁷ I understand that these elements are also explicitly and materially dependent on the modified OpenJDK source code.¹²⁸ I further understand that these elements would have no separate purpose and would not be functional without the modifications that Google has made to the

¹²⁵ A. Hall’s Opening Report, at ¶118.

¹²⁶ Expert report of Professor Doug Schmidt, February 8th, 2016.

¹²⁷ *Id.*

¹²⁸ *Id.*

OpenJDK source code.¹²⁹ As discussed above, Section 2 of GPLv2 requires not only the modified Program to be distributed under the terms of the GPLv2, but also the entire distributed work containing the modified Program, under the terms of the GPLv2. Given the relationships among the modified OpenJDK source code and the other elements of the Android stack discussed above, it is my opinion that there is a significant risk that source code for the application frameworks, the applications, and the Bionic libraries may need to be distributed using the GPLv2-CE or the GPLv2 license when Google publishes (distributes) the Android source code for OEMs to use.

171. The Hardware Abstraction Layer (“HAL”) is another element of the Android stack that is distributed together with modified OpenJDK code. I understand that the HAL is specifically intended and designed to function as part of the whole Android stack.¹³⁰ I further understand that the HAL would have no independent use outside of the Android platform, and that Android heavily relies on the modified OpenJDK-based libraries.¹³¹ Moreover, I understand that the HAL is explicitly and materially dependent on the Bionic library, which as I previously discussed, may also be subject to the requirements of the GPLv2-CE.¹³² As discussed above, Section 2 of GPLv2 requires that “the distribution of the whole must be on the terms of this License.” The only exception to this requirement is if the “independent and separate works” are distributed as separate works—not when they are included in the work. Given the relationship between the HAL and the rest of the Android stack, it is my opinion that the HAL may not be an “independent and separate work” and that, since it is distributed within the Android stack, there is significant risk that it too may need to be distributed under the terms of the GPLv2 or GPLv2-CE license in order to ensure compliance with the license when Google distributes its source code for OEMs to use.

172. Based on my understanding that Google distributes the Android stack as a whole work via the internet and in *source code* form, I think it likely that the Classpath Exception

¹²⁹ *Id.*

¹³⁰ *Id.*

¹³¹ *Id.*

¹³² *Id.*

does not exempt Google from the requirements of GPLv2 because Google is not distributing an executable. .

173. It should be noted I do not reach the same conclusion with respect to applications that Android application developers may create or have created by merely linking to the Android stack. My understanding from Dr. Schmidt is that Android application developers do not copy the 37 Java API packages into the programs that they distribute, they do not modify those API packages, and they do not distribute those API packages (which come preloaded on Android devices).¹³³ I understand that application developers invoke methods that use library names without copying the SSO (Structure, Sequence and Organization) embodied in the declaring code of the Java API packages.¹³⁴ Because these application developers do not engage in copying, distribution and modification of “the Program” (which Section 0 states are the only activities covered by the GPLv2), their activities are not governed by the provisions of the GPLv2-CE license.

B. Google’s Proposed Use of OpenJDK Could Expose Parts of OEMs’ Software to GPLv2 Obligations

174. It is my opinion that OEMs are permitted under GPLv2-CE to use the code they receive from Android—even if under GPLv2-CE—to form an executable by linking the GPLv2-CE code to their own independent modules, create a resulting executable, and then distribute that executable under the terms of their choice. However if OEMs *modify* any elements that are subject to the GPLv2-CE license, they risk being required to distribute those modifications (and the resulting whole work) under the terms of the GPLv2-CE in order to comply with the license.

175. I understand from Dr. Schmidt’s report that an analysis of OEM Android devices indicates that OEMs have modified the 37 Java API packages that Google now has made available under GPLv2-CE.¹³⁵ I further understand that OEMs have made modifications to the Application Framework Layer, the Application layer, and the HAL (as shown in the Android stack diagram).¹³⁶

¹³³ *Id.*

¹³⁴ *Id.*

¹³⁵ *Id.*

¹³⁶ *Id.*

176. Consistent with my explanation above, it is my opinion that there is a significant risk that OEMs would have to publish, in source code form, the modifications that they make to some elements of the Android stack when distributing their versions of Android using OpenJDK-based code. First, OEMs’ modifications of the 37 Java API packages would fall within the scope of Section 2 of GPLv2-CE. I understand that OEMs modify the Program itself, therefore Sections 2 and 3 require that the modified Program be distributed in source code form. Second, if (as discussed above) other elements of the Android stack, such as the application frameworks, the Bionic libraries, and the applications are subject to GPLv2-CE when distributed by Google (and therefore when modified by OEMs), *those* modified Programs would similarly have to be distributed in source code form. In either case, the Classpath Exception would not exempt OEMs from GPLv2 obligations because OEMs would be modifying the GPLv2 licensed Programs themselves — in other words, OEMs do not simply link to an independent module to create an executable as contemplated by the Classpath Exception.

177. Furthermore, I understand that OEMs typically use device drivers they develop and link into the HAL to differentiate their products from other Android-based devices, and those drivers are explicitly and materially dependent on both the HAL and the Bionic libraries.¹³⁷ If, as discussed above, either the HAL or the Bionic libraries are subject to GPLv2-CE, it is my opinion that because of the “whole work” provisions in Section 2, Android OEMs would risk needing to release the source code for these drivers in order to be compliant with the terms of the GPLv2 when the device driver code is distributed.

C. Hall Fails to Show that Google’s Proposed Combination is Permissible

178. In his report, Mr. Hall includes what he purports to be a chart describing commercial products that include OpenJDK code in closed-source products.¹³⁸ Mr. Hall thereafter concludes that it is permissible to use OpenJDK-based software in otherwise closed-source products. However, Mr. Hall’s chart merely lists what he claims are commercial products and references the license notices for those products.¹³⁹ Mr. Hall does

¹³⁷ *Id.*

¹³⁸ A. Hall’s Opening Report at ¶114.

¹³⁹ I note that this chart is not entirely accurate. For example, the Azul website featuring Zulu—one of the products cited by Mr. Hall as containing OpenJDK code in a closed-source commercial product—includes a clear notice stating that “Zulu is **100% open source** product and free to use and redistribute”. In other words, it is not a “closed- source” commercial product as Hall claims. See <https://www.azul.com/products/zulu/> (last visited Feb. 7, 2016).

not purport to analyze any of these products to confirm that OpenJDK code is actually used, to determine where and how OpenJDK code is used, to analyze whether the use is compliant with GPLv2-CE, or to examine whether the combination of licenses is compatible. In other words, Mr. Hall has done nothing to actually show that the use of OpenJDK code in these products (if it is in fact used in the products) is permissible.

179. As I have discussed throughout this report, compliance with a FOSS license is a case-specific, fact-intensive inquiry. Simply noting that some companies purport to use FOSS code in a particular way is not the same as saying that such use is permitted or license compliant. In *this* particular case, as noted above, Google's incorporation of modified OpenJDK-based code in the master branch of Android poses a significant risk that additional elements of the Android stack (including modifications made by OEMs) would be subject to the requirements of GPLv2-CE, meaning that Android's proposed combination of Apache and GPLv2-CE code would not be compatible.

XIV. Conclusion

180. GPL licenses are strong copyleft licenses that offer significant benefits in encouraging collaborative development of software and leveraging code already written (i.e. not reinventing the wheel). However, depending on the circumstances of a particular implementation, an organization seeking to include GPL-licensed code in an otherwise closed-source product may face significant risks in doing so, in that compliance with the GPL license could require that additional source code be made publicly available.

181. When Google first announced and launched Android, it appears Google was wary of including GPL-licensed code in the Android stack due to the potential copyleft implications of GPL licenses, particularly because Google anticipated that its OEMs would want to make proprietary modifications to Android and retain those modifications as closed-source.

182. My understanding of Google's proposed use and modification of the GPLv2-CE-licensed OpenJDK-based code in Android leads me to believe that there are significant risks that this use and modification could require other parts of the Android stack to be licensed under GPLv2. At least because Google distributes Android to its OEM partners in source code form, the Classpath Exception (which only applies to the distribution of executables) does not provide a safe harbor against the copyleft effects of GPLv2. Further, OEMs seeking to modify and thereafter distribute parts of the Android stack licensed under

GPLv2-CE would then be at significant risk of needing to release their own proprietary code in source code form to third parties under the GPLv2-CE license.

Executed on the 8th of February, 2016 in New York, New York.

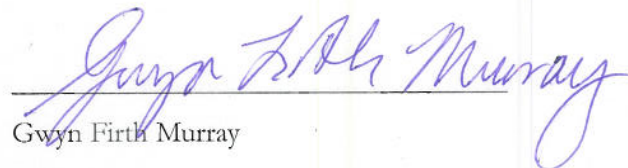

Gwyn Firth Murray

TABLE OF APPENDICES

Appendix	Title
A	Materials Considered
B	List of FOSS-Related Publications
C	List of FOSS-Related Speaking Engagements
D	CV
E	“Open Source” Criteria

Oracle America, Inc. v. Google Inc.

Case No. 10-03561 WHA

Appendix A - Materials Considered

PUBLIC SOURCES

Andreas Gal, Oracle sinks its claws into Android, January 5, 2016. http://andreasgal.com/2016/01/05/oracle-sinks-its-claws-into-android
Andrew Hall, Open Source Licensing And Business Models, All Things Open 2014. https://www.youtube.com/watch?v=OhQD6WD_KYE .
Andrew Miklas, Linksys WRT54G and the GPL, June 7, 2003. https://lkml.org/lkml/2003/6/7/164
Andrew Nusca, Best Buy, Samsung, Westinghouse, 11 more named in GPL lawsuit, December 14, 2009. http://www.zdnet.com/article/best-buy-samsung-westinghouse-11-more-named-in-gpl-lawsuit
Andrew St. Laurent Understanding Open Source and Free Software Licensing, O'Reilly, 2004.
Android Developers Blog, Announcing the Android 1.0 SDK, release 1, September 23, 2008. http://android-developers.blogspot.in/2008/09/announcing-android-10-sdk-release-1.html
Android tablet GPL compliance. http://www.codon.org.uk/~mjb59/android_tablets
Andy Rubin, Where's my Gphone?, November 5, 2007. https://googleblog.blogspot.com/2007/11/wheres-my-gphone.html
Anne McCauley, Wind River Systems Analyst Day - Final, March 26, 2008. 3/26/08 Fin. Disclosure Wire 16:00:00
Azul Systems, Inc., Go Further with Zulu and Zulu Enterprise. https://www.azul.com/products/zulu
Beth Z. Shaw, Recent Lawsuits Reflect Open Source Software Users' Copyright Compliance Obligations, May 7, 2010. http://www.wlf.org/publishing/publication_detail.asp?id=2164
Bill Burke, Apache damaging to Open Source. http://bill.burkecentral.com/2010/05/19/apache-damaging-to-open-source , May 19, 2010.
Bradley M. Kuhn, Aaron Williamson, Karen M. Sandler, <i>A Practical Guide to GPL Compliance</i> , SOFTWARE FREEDOM LAW CENTER, August 26, 2008.
Brian Carver, <i>Share and Share Alike: Understanding and Enforcing Open Source and Free Software Licenses</i> , Berkeley Technology Law Journal Vol. 20 - Issue 1, January 2005. https://www.softwarefreedom.org/resources/2008/compliance-guide.html
Brian Kelly, Oil and Water, or Vinaigrette? Open Source Software May Pose Legal Risks for Commercial Users, Andrews Intellectual Property Litigation Reporter, November 11, 2003.
Brown Rudnick, Emerging Technologies Blow, License Revoked: Applying Section 4 of the GPL and the Lessons of the Best Buy to Google's Android. http://brownrudnick.com/blog/emerging-technologies/license-revoked-applying-section-4-of-the-gpl-and-the-lessons-of-best-buy-to-googles-android/
Bruce Perens, Deciphering the war on open source, March 8, 2002. http://archive.linuxgizmos.com/bruce-perens-deciphering-the-war-on-open-source
BusyBox - GPL Lawsuits. http://www.liquisearch.com/busybox/gpl_lawsuits
Cade Metz, Google open source guru: 'Why we ban the AGPL', March 31, 2011. http://www.theregister.co.uk/2011/03/31/google_on_open_source_licenses
CBS/AP, Google I/O: Android TV, smart cars, wearables, and more, June 25, 2014. http://www.cbsnews.com/news/google-io-android-tv-smart-homes-wearables-and-more
Charles Babcock, Extreme Networks Latest Target of GPL Enforcement, July 22, 2008. http://www.informationweek.com/software/operating-systems/extreme-networks-latest-target-of-gpl-enforcement/d/d-id/1070196?
Charles J. Murray, Wind River nabs open-source, DSP software firms, April 5, 2001. http://www.eetimes.com/document.asp?doc_id=1143269
Christian Nandan, Open Source Licensing: Virus or Virtue?, 10 Tex. Intell. Prop. L.J. 349 (2002).
CommwebNews.com, Free Software Group Sues Cisco For Open Source Violations FSF claims the company failed to make source code for its routers to customers, December 11, 2008.
ComputerWeekly.com, Write Once, run anywhere? http://www.computerweekly.com/feature/Write-once-run-anywhere
Daniel Lyons, Linux's Hit Men, October 14, 2003. http://www.forbes.com/2003/10/14/cz_dl_1014linksys.html
David McGowan, Legal Implications of Open Source Software, 2001 U. Ill. L. Rev. 241 (2001).

Oracle America, Inc. v. Google Inc.

Case No. 10-03561 WHA

Appendix A - Materials Considered

David McGowan, The Tory Anarchism of F/OSS Licensing, 78 U. Chi. L. Rev. 207 (2011).
Eben Moglen, Enforcing the GNU GPL, September 10, 2001.
Eclipse Public License Version 1.0. https://www.eclipse.org/org/documents/epl-v10.html
Emil Protalinski, Google confirms next Android version will use Oracle's open-source OpenJDK for Java APIs, December 29, 2015. http://venturebeat.com/2015/12/29/google-confirms-next-android-version-wont-use-oracles-proprietary-java-apis/
Eric S. Raymond, The Cathedral & the Bazaar, Musings on Linux and Open Source by an Accidental Revolutionary, Tim O'Reilly, 1999.
Erick Schonfeld, Breaking: Google Announces Android and Open Handset Alliance, November 5, 2007. http://techcrunch.com/2007/11/05/breaking-google-announces-android-and-open-handset-alliance
FD Wire, Google, T-Mobile, HTC, Qualcomm, Motorola to Discuss New Open Platform for Mobile Devices- Final, November 5.
Free Software Foundation, Various Licenses and Comments about Them
GNU Classpath, Classpath License, July 30, 2009. http://www.gnu.org/software/classpath/license.html
GNU General Public License Version 2, June 1991. https://www.kernel.org/pub/linux/kernel/COPYING
GNU General Public License version 2.0, April 12, 2014. http://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html
GNU Lesser General Public License, version 2.1. http://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html
GNU Operating System, Frequently asked Questions about the GNU Licenses, January 21, 2016. http://www.gnu.org/licenses/gpl-faq.en.html#GPLRequireSourcePostedPublic
GNU Project - Free Software Foundation, About The GNU Project. https://www.gnu.org/gnu/thegnuproject.html
GNU, Initial Announcement. https://www.gnu.org/gnu/initial-announcement.html
GNU, Overview of the GNU System. http://www.gnu.org/gnu/gnu-history.en.html
Google Git, AccessControlContext.java, https://android.googlesource.com/platform/libcore/+master/ojiluni/src/main/java/java/security/AccessControlContext.java
Google Git, AddressCache.java, (2010). https://android.googlesource.com/platform/libcore/+master/luni/src/main/java/java/net/AddressCache.java
GPL Enforcement Cases. https://wiki.fsfe.org/GPL%20Enforcement%20Cases
GPL-Violations, About the GPL-Violations.org Project. http://gpl-violations.org/about/
Greg R. Vetter, <i>Commercial Free and Open Source Software: Knowledge Productions, Hybrid Appropriability, and Patents</i> , 77 FORDHAM L. REV. 2087 (2009).
Groklaw, Software Freedom Law Center Launched, February 1, 2005. http://www.groklaw.net/articlebasic.php?story=20050201060017590
Groklaw, The Oracle v. Google Trial Exhibits - Can You Help List Them?, May 19, 2012. http://www.groklaw.net/articlebasic.php?story=20120519115125315
Heather Meeker, Open Source and the Legend of Linksys, June 28, 2005. http://www.linuxinsider.com/story/43996.html?%20wlc=1300413418
Heather Meeker, Open Source and the Secrets Commando Due Diligence. 43-WTR Tex. J. Bus. L. 561 (2009).
History of Java programming language, http://www.freejavaguide.com/history.html
History of Java Technology. http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html
Interview with Sleepycat President and CEO, Michael Olsen, October 29, 2001, http://www.winterspeak.com/2001/10/interview-with-sleepycat-president-and.html
Jason Chen, Googler Developer Advocate, video re Releasing Android under License at Google I/O Conference, 2008. https://www.youtube.com/watch?v=x1ZZ-R3p_w8

Oracle America, Inc. v. Google Inc.

Case No. 10-03561 WHA

Appendix A - Materials Considered

Jennifer Buchanan O'Neill, Christopher Gasper, What Can Decisions by European Courts Teach Us About the Future of Open-Source Litigation in the United States, 38 AIPLA Q.J. 437 (2010).
John Edwards, Infoworld, Does GPL still matter?, August 10, 2009. http://www.infoworld.com/article/2631562/software-licensing/does-gpl-still-matter-.html
John Rose, with Android and Dalvik at Google I/O, May 31, 2008.
John Timmer, BusyBox takes out bankrupt opponent in GPL lawsuit, August 5, 2010. http://arstechnica.com/information-technology/2010/08/court-rules-gpl-part-of-a-well-pleaded-case
Jon Brodtkin, Android violates Linux license, experts claim, March 17, 2011.
Jon Brodtkin, Most Android, iPhone apps violate open source rules- Mobile app developers fail to comply with Apache, GPL requirements, survey finds, 2011 WLN 5025874, March 9, 2011.
Juan Carlos Perez, Update: Google details highly anticipated 'Android' mobile platform, 2007 WLN 21857371, November 5.
Karl Kogel, Producing Open Source Software, How to Run a Successful Free Software Project, O'Reilly, 2006.
Kevin Gard, Jen Salyers, Open Source- Friend or Foe?, 18 DCBA Brief 22 (2006).
Kevin J. O'Brien, In open source, an unexpected trap License rules snare even big tech firms, December 10, 2005. https://www.highbeam.com/doc/1P1-116284909.html
Larry Rosen, Open Source Licensing: Software Freedom and Intellectual Property Law, 2005.
Lawrence Lessig, Free Software Free Society: selected essays of Richard M. Stallman, GNU Press, 2002.
Lawrence Rosen, Open Source Licensing, Software Freedom and Intellectual Property Law, Prentice Hall Professional.
Marcel de Vries, Best Practices for Using Open Source Software in the Enterprise, TechEd Europe 2014. https://www.youtube.com/watch?v=HOCPD3v-Tx8
Mark Radcliffe, Thomas Jansen, Hannes Meyle, The Fantec decision: German Court holds distributor responsible for FOSS Compliance. July 30, 2013, https://opensource.com/law/13/7/fantec-german-foss-compliance
Mark Wielaard, Toward a free Java, May 24, 2006, https://lwn.net/Articles/184967
MarketWatch, Oracle to buy Sun Microsystems for 5.6 billion, April 20, 2009, http://www.marketwatch.com/story/oracle-buy-sun-microsystems-74
Marketwire, OpenLogic Scanning Data Reveals OSS Developers Choose GPL, Enterprises Prefer Apache, May 16, 2011. http://www.marketwired.com/press-release/openlogic-scanning-data-reveals-oss-developers-choose-gpl-enterprises-prefer-apache-1514793.htm
Martin Fink, The Business and Economics of Linux and Open Source, Prentice Hall PTR, 2003.
Maxim Tsotsorin, Open Source Software Compliance: The Devil is Not So Black as He is Painted, Santa Clara High Technology Law Journal Vol. 29 Issue 3, (2012).
Michael Lerner, Audrey Haroz Reed, Philip Porter, Open Source Software and Risks Associated with its Use in a Commercial Environment, 9 No. 3 Cyberspace Law. 7, (2004).
Michael Morgan, The Cathedral and the Bizarre: A Examination of the "Viral" Aspects of the GPL, 27 J. Marshall J. Computer & Info. L. 349 (2010).
Mikko Valimaki, Dual Licensing in Open Source Software Industry, January 2003, Published in Systemes d'Information et Management.
Nancy Gohring, Google set to release Android source code, October 21, 2008.
Natasha Lomas, Android Now Has 1.4 Billion 30-Day Active Users Globally, September 29, 2015. http://techcrunch.com/2015/09/29/android-now-has-1-4bn-30-day-active-devices-globally .

Oracle America, Inc. v. Google Inc.

Case No. 10-03561 WHA

Appendix A - Materials Considered

Neon Kelly, Google goes mobile with Android plan, November 15, 2007.
Open Handset Alliance- Alliance Members, http://www.openhandsetalliance.com/oha_members.html
Open Handset Alliance, Industry Leaders Announce Open Platform for Mobile Devices, November 5, 2007. http://www.openhandsetalliance.com/press_110507.html
Open Invention Network, Home Page, http://www.openinventionnetwork.com
Open JDK, JDK 8, March 18, 2014, http://openjdk.java.net/projects/jdk8
Open Source Initiative, History of the OSI, http://opensource.org/history
Open Source Initiative, Licenses by Name, http://www.opensource.org/licenses/alphabetical
Open Source Initiative, The BSD 2-Clause License, http://www.opensource.org/licenses/bsd-license.php
Open Source Initiative, The Open Source Definition, March 22, 2007, https://opensource.org/osd
Open Sources, Introduction, Chris DiBona, Sam Ockman, Mark Stone, Introduction, Open Sources: Voices from the Open Source Revolution, (O'Reilly 1999). http://www.oreilly.com/openbook/opensources/book/intro.html
OpenJDK, GNU General Public License, version 2, with the Classpath Exception, June 1991.
OpenJDK Classpath Exception. http://openjdk.java.net/legal/gplv2+ce.html
OpenJDK, OpenJDK FAQ, December 18, 2010. http://openjdk.java.net/faq
OpenLogic, Busting the Myth of GPL Dominance: Apache Rules!, September 27, 2007. http://www.openlogic.com/resources/enterprise-blog/archive/busting-the-myth-of-gpl-dominance-apache-rules
Oracle Announces Java 8, March 25, 2014. http://www.oracle.com/us/corporate/press/2172618
Oracle Buys Sun, April 20, 2009. http://www.oracle.com/us/corporate/press/018363
Oracle, Introducing Java 7 Webcast: Moving Java Forward, July 7, 2011. http://www.oracle.com/webapps/events/ns/EventsDetail.jsp?p_eventId=134208&src=7299693&src=7299693&Act=18
Patrick Brady, Anatomy and Physiology of an Android, Google I/O Conference 2008. https://www.youtube.com/watch?v=G-36noTCaiA
Paul Krill, Open source Java for Android? Don't bet on it, November 20, 2012. http://www.javaworld.com/article/2078666/mobile-java/open-source-java-for-android--don-t-bet-on-it.html
Paul Vixie, Software Engineering, Open Sources: Voices from the Open Source Revolution, (O'Reilly 1999). http://www.oreilly.com/openbook/opensources/book/vixie.html
Plaintiffs' Memorandum of Law in Support of Their Motion for Preliminary Injunction Against Defendants Best Buy, Co., Inc. and Phoebe Micro, Inc., January 31, 2011. http://www.groklaw.net/pdf3/SFCvBestBuy-164.pdf
Rachel Stern, Erik Kane, Emily Prudente, Open for Business: What Corporate Counsel Need to Know in the Intricate World of Open Source Code, 26 No. 10 ACC Docket 40 (2008).
Richard Stallman, The GNU GPL and the American Way, April 12, 2014. http://www.gnu.org/philosophy/gpl-american-way.en.html
Richard Stallman, Why Open Source Misses the Point of Free Software, January 1, 2016. http://www.gnu.org/philosophy/open-source-misses-the-point.html
Robert Eckstein, James Gosling on Open Sourcing Sun's Java Platform Implementations, Part 2, November 2006. http://www.oracle.com/us/technologies/java/gosling-os2-qa-136546.html
Robert Gomulkiewicz, Open Source License Proliferation: Helpful Diversity or Hopeless Confusion, 30 Wash. U. J.L. & Pol'y 261 (2009).

Oracle America, Inc. v. Google Inc.

Case No. 10-03561 WHA

Appendix A - Materials Considered

Robert Young; <i>"Giving It Away: How Red Hat Software Stumbled Across a New Economic Model and Helped Improve an Industry"</i> , Open Sources: Voices from the Open Source Revolution, (O'Reilly 1999). http://www.oreilly.com/openbook/opensources/book/young.html
Savio Rodrigues, Activiti BPM project questions value of LGPL GNU license, May 19, 2010. http://www.infoworld.com/article/2626845/technology-business/activiti-bpm-project-questions-value-of-lgpl-gnu-license
Serdar Yeggulalp, Why GPL still gives enterprises the jitters, July 7, 2014. http://www.infoworld.com/article/2608340/open-source-software/why-gpl-still-gives-enterprises-the-jitters.html
SGI Expands Server Operating System Strategy: Company Adds Linux And Enhances IRIX, May 17, 1999. http://www.prnewswire.com/news-releases/sgi-expands-server-operating-system-strategy-company-adds-linux-and-enhances-irix-74548417.html
Software Freedom Conservancy. https://sfconservancy.org/about/
Software Freedom Law Center, BusyBox Developers Agree to End GPL Lawsuit Against Verizon- Good Faith Discussions Result in Dismissal of Copyright Infringement Case, March 17, 2008. http://www.softwarefreedom.org/news/2008/mar/17/busybox-verizon
Software Freedom Law Center, BusyBox Developers and High-Gain Antennas Agree to Dismiss GPL Lawsuit- Good Faith Discussions Result in Closing of Copyright Infringement Case, March 6, 2008. http://www.softwarefreedom.org/news/2008/mar/06/busybox-hga
Software Freedom Law Center, BusyBox Developers and Monsoon Multimedia Agree to Dismiss GPL Lawsuit- Productive and Amicable Negotiations Lead to Settlement of Copyright Infringement Case, October 30, 2007. http://www.softwarefreedom.org/news/2007/oct/30/busybox-monsoon-settlement/
Software Freedom Law Center, BusyBox Developers and Supermicro Agree to End GPL Lawsuit- Good Faith Discussions Result in Dismissal of Copyright Infringement Case, July 23, 2008. http://www.softwarefreedom.org/news/2008/jul/23/busybox-supermicro
Software Freedom Law Center, BusyBox Developers and Xterasys Corporation Agree to Settle GPL Lawsuit, December 17, 2007. http://www.softwarefreedom.org/news/2007/dec/17/busybox-xterasys-settlement
Software Freedom Law Center, Motion Against Westinghouse Digital Electronics in GPL Compliance Lawsuit, June 7, 2010. https://www.softwarefreedom.org/news/2010/jun/07/motion-against-westinghouse-digital-electronics-gp
Software Freedom Law Center, SFLC Provides Ongoing GPLv3 Legal Support to FSF, January 19, 2008. http://www.softwarefreedom.org/news/2008/jan/19/gplv3-support
Stack Overflow, Java - Do I need to open my source code if I use OpenJDK as JVM? http://stackoverflow.com/questions/4143635/do-i-need-to-open-my-source-code-if-i-use-openjdk-as-jvm/4443281#4443281
Stephen O'Grady, What are the Most Popular Open Source Licenses Today?, http://redmonk.com/sogrady/2014/11/14/open-source-licenses
Steven Vaughan-Nichols, The fall of GPL and the rise of permissive open-source licenses, December 16, 2014. http://www.zdnet.com/article/the-fall-of-gpl-and-the-rise-of-permissive-open-source-licenses/
Steven Vaughan-Nichols, First Lawsuit Filed Over Alleged GPL Violation, September 21, 2007. http://www.extremetech.com/extreme/79448-first-lawsuit-filed-over-alleged-gpl-violation
Sun Fulfills Promise of Open and Free Java Technology and Releases Java SE Platform to OpenJDK Community, May 8, 2007. https://web.archive.org/web/20070510070408/http://www.sun.com/aboutsun/pr/2007-05/sunflash.20070508.3.xml
Terminology Wars-A Web Content Analysis, October 27, 2004. http://www.catb.org/~esr/writings/terminology/#results

Oracle America, Inc. v. Google Inc.

Case No. 10-03561 WHA

Appendix A - Materials Considered

Texas Instruments Incorporated, Texas Instruments further expands support for the Open Handset Alliance and open source community, October 28, 2008. http://newscenter.ti.com/index.php?s=32851&item=126078
The Apache Software Foundation, Apache License v2.0 and GPL Compatibility, http://www.apache.org/licenses/GPL-compatibility.html
The Apache Software Foundation, Licensing of Distributions, http://www.apache.org/licenses
The Open Source Alternative: Understanding Risks and Leveraging Opportunities. Heather J. Meeker. ISBN: 978-0-470-19495-9, February 2008.
Thomas Alspaugh, Hazeline Asuncion, Walt Scacchi, The Challenge of Heterogeneously Licensed Systems in Open Architecture Software Ecosystems. http://www.ics.uci.edu/~wscacchi/Papers/New/SEAMBNSI-AlspaughAsuncionScacchi.pdf
Tim O'Reilly, Hardware, Software, and Infoware, Open Sources: Voices from the Open Source Revolution, (O'Reilly 1999). http://www.oreilly.com/openbook/opensources/book/tim.html
Top 20 Open Source Licenses, https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses
Van Lindberg, Intellectual Property and Open Source A Practical Guide to Protecting Code, O'Reilly, 2008.
What is Copyleft?, October 3, 2015. http://www.gnu.org/copyleft/copyleft.html
What is free software?, January 1, 2016. http://www.gnu.org/philosophy/free-sw.en.html
Wireless Europe's Daily Bulletin, 2007 WLNLR 23836464, Report- Google's Android May Justify Vast Investment in Platform Development, December 3, 2007.
XDA, OEMs and GPL Compliance, July 2, 2013. http://www.xda-developers.com/oems-and-gpl-compliance
CODES
17 USC § 101 Definitions
TRIAL EXHIBITS
TX0230, Email from Andy Rubin to Bob Lee re New OpenJDK Community Technology Compatibility Kit License (TCK), dated August 11, 2007.
EXPERT REPORTS
Opening Expert Report of Professor Douglas C. Schmidt, Ph.D., dated January 8, 2016
Opening Expert Report of Andrew Hall, dated January 8, 2016
Excerpt of Expert Report of Dr. Owen Astrachan (Paragraph 264), dated January 8, 2016

APPENDIX B – FOSS-Related Publications

- Getting With the Program: A Guide for Lawyers Working with Free and Open Source Software in the Enterprise; Paper presented at “Implementing Open Source Software in the Enterprise” conference (Law Seminars International, Seattle, WA 2005).
- Free and Open Source Software: An Introduction. Co-authored with Michael A. Duncheon of Hanson, Bridgett, Marcus, Vlahos & Rudy, LLP (January 2006).
- Bringing FOSS Into the Enterprise: Thoughts on Best Practices. Published in Oregon Intellectual Property Network.
- Attribution Requirements in Free and Open Source Software Licensing, in The Computer & Internet Lawyer, Vol 25, No. 5, May 2008
- Open Source: A Challenge Worth Meeting, in the California Business Law Practitioner, Vol 24, No. 4, Fall 2009
- Categorization of Open Source Licenses: More Than Just Semantics, in The Computer & Internet Lawyer, Vol 26, No 1, January 2009

APPENDIX C – FOSS-Related Speaking Engagements

- “Elimination of Bias in the Engineering and Legal Professions”, at Practicing Law Institute, Open Source and Free Software 2015: Benefits, Risks and Challenges, December 2015, San Francisco, CA.
- Free and Open Source Software (“FOSS”) Licensing Overview, at Institute of Electrical and Electronics Engineers Meeting, San Jose, CA, February 2015.
- Free and Open Source Software (“FOSS”) Licensing Overview, at Bay Area Open Source Meetup, Palo Alto, CA, January 2015.
- Ethics Issues in the Practice of Open Source Law, at Practicing Law Institute, Open Source and Free Software 2014, December 2014, San Francisco, CA
- Ethics: Conflict and Cooperation in Open Source Projects, at Practicing Law Institute Open Source and Free Software 2013, December 2013, San Francisco, CA
- “Ethical Implications of Open Source Strategies: Background Material?”, at Practicing Law Institute Open Source and Free Software 2012: Benefits, Risks and Challenges, December 2012, San Francisco, CA
- Speaker, Silicon Valley Linux Users’ Group, September 2012, San Jose, CA
- Co-Panelist, The Role of Emerging Technologies in Entrepreneurship, at “Entrepreneurship and the Law: Navigating the Legal Challenges Facing Growing Ventures” March 2011, at UC Davis School of Law, Davis, CA
- “Ethical Implications of Open Source Strategies: Background Material?”, at Practicing Law Institute Open Source and Free Software 2010: Benefits, Risks and Challenges, December 2010, San Francisco, CA
- “Ethical Implications of Open Source Strategies, at Practicing Law Institute Open Source and Free Software 2009: Benefits, Risks and Challenges in Today’s Economic Environment, December 2009, San Francisco, CA
- The Anatomy of an Open Source License, Categorization of open source licenses: it's more than just semantics, Open source software 2008: Benefits, Risks and Challenges for Software Users, Developers and Investors, at Practicing Law Institute, December 2008, San Francisco, CA
- Free and Open Source Software: What Is It? Where Did It Come From? How Do I Use It? Breakfast Briefing at Hoge, Fenton, Jones & Appel, San Jose, California. July, 2008

- Free and Open Source Software: How to Get Money and Make Money with FOSS. Evening presentation for The East Bay Innovation Group (eBIG), Pleasanton, CA. November 2008
- “The Anatomy of an Open Source License” at Open Source Software 2008: Benefits, Risks and Challenges for Software Users, Developers and Investors (Practicing Law Institute, San Jose, CA 2008)
- “Free and Open Source Software”, Brown Bag Briefing at Hoge, Fenton, Jones & Appel (San Jose, CA 2008)
- Panelist, Open Source Licensing Issues, at Open Source in Mobile Conference (San Francisco, CA 2008)
- “FOSS: What's New -- and Not New,” at California Cyberspace Committee Meeting (East Palo Alto, CA, February 2008)
- “FOSS: What’s New – and Not New”, at Fundamentals of Software Transactions (Silicon Valley Seminars, Palo Alto, CA 2008)
- “FOSS: What’s New and What’s Not New” (Cyberspace Committee of the California Bar, East Palo Alto, CA 2008)
- Co-presenter, The Legal Realities of Open Source Licenses, at Open Source Software 2007: Risks, Rewards and Practical Realities in the Corporate Environment (Practicing Law Institute, San Francisco, CA 2007)
- GPL Version 3.0: What’s New and What’s Not? (Intellectual Property Society, Sunnyvale, CA 2007)
- Panelist, Open Source Software: Practical Guidance in Locating Open Source Code and Determining the Best Policies to Deal With It (Law Seminars International, San Francisco, CA 2007)
- Panelist, Negotiating and Drafting IP License Agreements (Continuing Education of the Bar, San Francisco, CA 2007)
- Panelist, Open Source Software: Trends and Implications For The Licensing Community, at Managing the Evolving Deal (Licensing Executives Society Winter Meeting, San Francisco, CA 2007)
- Panelist, Myths Around Open Source Licensing (Licensing Executives Society, Palo Alto, CA 2006)
- Panelist, Legal Issues in Open Source (Association of Corporate Counsel, Burlingame, CA 2006)
- Speaker, Best Practices for Open Source Software (Event Co-hosted by Open Bar, Inc and Open Source Development Labs, Inc., and Lane Powell LLP, Portland, OR 2006)

- “To Ship or Not to Ship: What Are the Questions?” at Doing Business with Open Source Software Conference (Law Seminars International, San Francisco, CA 2006)
- “Free and Open Source Software in the Enterprise” at Understanding the Intellectual Property License (Practicing Law Institute, San Francisco, CA 2005)
- “Implementing Open Source Software in the Enterprise” (Law Seminars International, Seattle, WA 2005)
- Moderator, “Looking Before You Leap” Panel (Open Bar, Inc. and Wilson, Sonsini, Goodrich and Rosati, Palo Alto, CA 2004)

APPENDIX D - CV

GWYN FIRTH MURRAY

223 Laurel Avenue
Menlo Park, CA 94025
(650) 823-5864 (cell)
gwyn@mataulegal.com
www.mataulegal.com

GENERAL:

- Experienced attorney offering a wide range of cost-effective legal services to both start-up and established companies in the high tech and biotech industries
- “Early adopter” in the legal community with respect to free and open source software issues, with broad expertise in the area of Free and Open Source Software (FOSS) licensing and compliance
- Extensive expertise in software licensing, commercial sales and services, intellectual property management, and corporate and business strategy
- Managed all legal services for complex businesses encompassing hardware sales, software licensing, technical and professional services and operation of customer-facing web networks
- Drafted and negotiated a wide variety of commercial and licensing agreements, both in the U.S. and internationally
- As General Counsel and outside counsel, managed IP programs, litigation strategies and corporate and securities compliance for both public and private companies.

EMPLOYMENT:

MATAU LEGAL GROUP, Menlo Park, CA
Founder and Principal (2002 to present)

Independent attorney focused on (i) providing “outside general counsel” services for companies that, given their maturity and/or budget constraints, do not employ full-time internal counsel, (ii) advising on open source software licensing issues and compliance, and (iii) assisting client companies with structuring and negotiating a wide variety of commercial and licensing transactions in the United States and overseas, particularly in Latin America and Europe. Offer a wide range of cost-effective legal services to both privately and publicly-held companies in the high-tech and biotech industries, including general corporate and intellectual property advice, international expansion and sales; and assistance on human resources issues and real estate matters. Areas of particular specialty include open source/free software licensing, commercial operations domestically and overseas, information technology, proprietary software and biotechnology licensing, and assistance to companies providing consulting and/or professional services in the areas of open and closed-source software development.

KANISA INC. Cupertino, California
Vice President -- Legal Affairs and General Counsel (2001-2002)

Managed all legal matters of the company, including structuring and negotiating complex software licensing and services agreements with Fortune 100 companies; designing and implementing the company’s intellectual property strategy, including a robust patent and trademark portfolio; providing advice to management and the Board of Directors on legal strategy, corporate governance and securities matters; advising on merger and acquisition transactions; advising and training employees on employment, ethics and other matters; reviewing and advising on press and other external communications; negotiating real estate leases; managing the company’s litigation strategy; and selecting and managing outside counsel.

VA LINUX SYSTEMS. Fremont, California
Vice President of Legal Services (1999-2000)
Director of Legal Affairs

Managed all legal matters of a pioneer company in the open source software arena. Practice included managing \$2.5MM annual legal budget; providing advice to management and the Board of Directors on corporate governance and securities matters; advising on merger and acquisition transactions; structuring and implementing international operations and joint ventures; designing and implementing company policies; developing company's strategy on intellectual property management and open-source software development; advising and training employees on labor issues and other matters; drafting and negotiating commercial, services and marketing agreements; reviewing and advising on press and other external communications; negotiating real estate leases; supervising litigation matters; and selecting and managing legal staff and outside counsel.

SILICON GRAPHICS (SGI). Mountain View, California
Senior Legal Counsel, SGI Global Services (1998 – 1999)

Supervised worldwide legal affairs of SGI Global Services, a profitable business with \$700 million in annual revenues. Responsible for developing and implementing legal infrastructure supporting SGI's professional services and customer support businesses, including managing legal support staff, outside counsel and budget; leveraging field legal resources; creating and "rolling out" form professional and customer service contracts; training field legal staff in contract negotiation and intellectual property licensing issues; managing litigation and offering guidance on day-to-day professional services, customer support and supplier transactions. Served as lead lawyer on global, strategic services deals, including negotiation of licensing, joint venture, reseller, porting and joint marketing agreements with SGI's partners. Participated in SGI's transition to running its hardware on the Linux Operating System.

SILICON GRAPHICS (SGI). Mountain View, California
Legal Counsel, Americas International (1995-1998)

Managed legal affairs of SGI's subsidiaries and branch offices, as well as agreements with distributors, dealers, and private and government customers throughout Latin America and Canada. Worked closely with Regional and Country Managers in developing and implementing business and marketing strategies, personnel matters and negotiation of equipment leasing, real estate and construction contracts. Trained employees on subjects such as anti-corrupt practices, export rules, use of form contracts, and trademark usage. Managed relationships with outside counsel in each country, including budget planning and control. Coordinated the legal aspects of the transition of employees, facilities and corporate structure in the region in connection with the purchase of Cray Research by SGI.

ALZA CORPORATION. Palo Alto, California
Corporate Attorney (1994-1995)

Primary lawyer for ALZA's then newly formed sales and marketing group, and principal liaison to the Company's regulatory department. The lawyer responsible for ALZA's compliance with regulations relating to the manufacture, sales, marketing and distribution of pharmaceutical products, and with federal anti-kickback and antitrust requirements. Negotiated and drafted sales and distribution agreements with US-based, Asian and European companies. Advised on product complaint and product liability issues, employee training on legal and administrative law matters, and interviewing, hiring and managing outside FDA counsel.

APPLE COMPUTER. Cupertino, California
Counsel (1990 -1994)

Supported Apple's Management and Board of Directors on corporate governance, transactional and securities matters. Served as lead lawyer for financial transactions for the Company's Treasury Group,

including interest rate and currency swap transactions, equity option programs, and a \$500 million debt offering. Negotiated and drafted agreements for Apple's investments in, loans to, and associated product licenses with start-up software and hardware companies.

ARNOLD & PORTER. Washington, D.C.
Associate (1988-1990)

Worked on international trade and finance transactions, including general assistance on restructuring of Brazilian external debt and litigation of anti-dumping matters on behalf of the Asociacion Colombiana de Exportadores de Flores.

EDUCATION:

J.D. Stanford Law School, Stanford University. 1987.
Fellowship: Foreign Language and Area Studies (FLAS) Grant

M.A. Department of Latin American Studies, Stanford University. 1987.

B.A. Economics, Yale University. 1982.
Honors: Magna Cum Laude, Distinction in Economics

COMMUNITY INVOLVEMENT:

Open Bar, Inc. Palo Alto, California
Co-Founder and Member, Board of Directors (2003 – present)

Open Bar is a not-for-profit organization founded by legal professionals who seek to (i) develop clear information about the legal rights and responsibilities of software developers, legal professionals and users of software in the emerging arena of open source/free software; and (ii) educate software developers, legal professionals and the general public about the issues, rights and responsibilities associated with the development, use and distribution of open source/free software.

Idea Parlor, Inc. San Jose, California (www.ideaparlorg.org)
Co-Founder and Member, Board of Directors (2008 – present)

Idea Parlor is a not-for-profit organization established with the goal of nurturing ideas into inventions. The Idea Parlor reaches out to all individuals, groups and organizations; those new to inventing or not; whether independents or part of an organization/company: in all cases to help evolve creative ideas into tangible inventions. The Idea Parlor is a nurturing and fun environment dedicated to bringing whims and serious thoughts into the realm of potentially patentable inventions and to promote public awareness about the steps to identifying an invention and techniques to make it happen.

Bay Area Adoption Services (BAAS), Mountain View, California (www.baas.org)
Member, Board of Directors (2000 – 2005)

BAAS is a not-for-profit agency whose mission is placing orphaned and abandoned children with parents. BAAS has established a strong network of credible placement sources throughout the world. BAAS has united loving parents in the United States with children from around the world, including Asia, the Philippines, Central and South America, India, Eastern Europe, and the former Soviet Union.

Bar Memberships: State Bar of California, admitted 1987; District of Columbia Bar, admitted 1988

Languages: English, Spanish and Portuguese

Personal Interests: Beekeeping; organic vegetable and herb gardening; hiking; Latin dancing and stand-up comedy.

APPENDIX E – “Open Source” Definition

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Last modified, 2007-03-22

PROOF OF SERVICE BY KITEWORKS

I, José E. Valdés, am over the age of eighteen years old and not a party to the within-entitled action. My place of employment and business address is Orrick, Herrington & Sutcliffe LLP, 1000 Marsh Road, Menlo Park, California 94025.

On February 8, 2016, I served the following documents:

EXPERT REPORT OF GWYN FIRTH MURRAY

on the interested parties in this action by electronic service [Fed. Rule Civ. Proc. 5(b)] by electronically mailing a true and correct copy, pursuant to the parties agreement, to the following email addresses:

DALVIK-KVN@kvn.com
JCooper@fbm.com
gglas@fbm.com

I declare under penalty of perjury under the laws of the State of California and the United States that the foregoing is true and correct.

Executed on February 8, 2016, at San Francisco, California.

/s/ José E. Valdés
José E. Valdés